Desarrollo de un Modelo Híbrido usando Modelos de Aprendizaje Profundo para la Recuperación de Información Multi-Modal en Texto e Imágenes

Miler Diaz Zevallos

Orientador: Dr. José Eduardo Ochoa Luna

Jurado:
Dr. Omar U. Florez Choque – Intel Labs – USA
Dr. Guillermo Cámara Chávez – Universidade Federal de Ouro Preto – Brasil
Dr. Yván Túpac Valdivia – Universidad Católica San Pablo – Perú
Dr. Alex Cuadros – Universidad Católica San Pablo – Perú

Tesis presentada al
Centro de Investigación e Innovación en Ciencia de la Computación (RICS)
como parte de los requisitos para obtener el grado de
Maestro en Ciencia de la Computación.

Universidad Católica San Pablo – UCSP
Mayo de 2017 – Arequipa – Perú

El presente trabajo está dedicado a mi madre y mi abuela por su constante confianza y apoyo durante todo el proceso de desarrollo de la presente tesis, a mi familia por apoyarme en cada una de las decisiones que tomo, a mis compañeros y docentes de la universidad por la ayuda brindada en la realización de este proyecto y a Dios por su apoyo y guía.

Abreviaturas

IR *Information Retrieval*

AI *Artificial Intelligence*

ML *Machine Learning*

DL *Deep Learning*

ANN *Artificial Neural Network*

DNN *Deep Neural Network*

RBM *Restricted Boltzmann Machine*

AE *AutoEncoder*

DAE *Deep Autoencoder*

CNN *Convolutional Neural Network*

CD *Constractive Divergence*

MAP *Mean Average Precision*

SAE *Stacked Autoencoder*

TDSN *Tensor Deep Stacking Network*

DSN *Deep Stacking Network*

Agradecimientos

Agradezco primeramente a Dios por ayudarme y haberme dado la suficiente fuerza y sabiduría para poder terminar este proyecto durante los 2 años de estudio, que duró el periodo de la maestría.

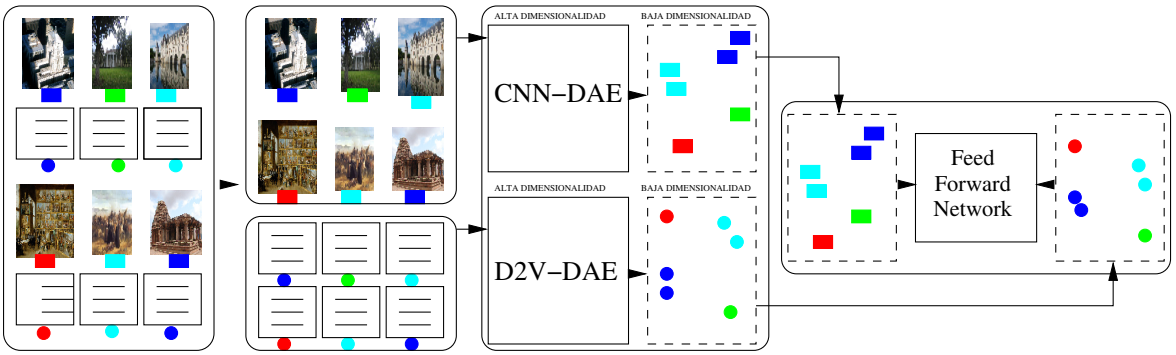
Agradezco a mi madre, mi abuela y mi familia por el apoyo que me dieron en cada una de las decisiones que tomo.

Deseo agradecer de manera especial al Consejo Nacional de Ciencia, Tecnología e Innovación Tecnológica (CONCYTEC) y al Fondo Nacional de Desarrollo Científico, Tecnológico e Innovación Tecnológica (FONDECYT-CIENCIACTIVA), que mediante Convenio de Gestión UCSP-FONDECYT Nº 011-2013, han permitido la subvención y financiamiento de mis estudios de Maestría en Ciencias de la Computación en la Universidad Católica San Pablo (UCSP).

Agradezco de forma muy especial a mi orientador Dr. José Eduardo Ochoa Luna por haberme guiado en esta tesis.

Agradezco a la universidad, por haberme cobijado y brindado la formación que ahora me permitirá ayudar a construir una mejor sociedad.

Abstract



Currently the use of Deep Learning models in many areas of research is showing excellent results, the area of Information Retrieval is one of them. Within this area, there is a task that is Information Retrieval in multiple modalities. The main objective of this task is to project data from different modalities within a common semantic space or create a model to establish a relationship between these spaces. In this research we propose two intra-modal hybrid models for dealing with images and texts respectively and the development of a model to establish a relationship between the two modalities. The results will be evaluated on several data sets used in the state-of-art to validate the performance of the general model.

Keywords: Multi-modal information retrieval, Deep learning, Feature extraction.

Resumen

Actualmente el uso de los modelos de Aprendizaje Profundo en muchas áreas de investigación esta demostrando excelentes resultados, el área de Recuperación de Información es una de ellas. Dentro de esta área existe una tarea que es la Recuperación de Información en múltiples modalidades. El objetivo principal de esta tarea es proyectar datos de diferentes modalidades dentro de un mismo espacio semántico o crear un modelo para establecer una relación entre estos espacios. En esta investigación se propone dos modelos híbridos intra-modales para tratar con imágenes y textos respectivamente y la elaboración de un modelo para establecer una relación entre ambas modalidades utilizando modelos de Aprendizaje Profundo. Los resultados serán evaluados en varios conjuntos de datos utilizados en el estado del arte para validar el rendimiento del modelo general.

Palabras clave: Recuperación de información multi-modal , Aprendizaje profundo, Extracción de características.

Índice general

Índice de cuadros

Índice de figuras

Capítulo 1

Introducción

En los últimos años, el avance tecnológico ha conducido a la producción de innumerables archivos digitales (documentos, imágenes, multimedia, entre otros), los cuales se encuentran almacenados en repositorios de datos. Cada año, el tamaño de estas colecciones se incrementa considerablemente (Vikram, 2015). Las estadísticas elaboradas sobre la red social *Twitter* revelan que 500 millones de *tweets*, en promedio, son enviados por día [1], mientras que en *Facebook* se reporta que 300 millones de fotos son almacenadas cada día [2]. Para facilitar la búsqueda de información relevante sobre grandes cantidades de datos y sobrellevar este rápido crecimiento, se necesita del desarrollo de nuevos sistemas de recuperación de información, los cuales deben de mejorar su rendimiento al trabajar a gran escala.

El proceso de Recuperación de Información o *Information Retrieval* (IR), consiste en encontrar un objeto o material (generalmente documentos), de naturaleza no estructurada (generalmente texto o páginas web), para satisfacer una necesidad de información (proveniente de un usuario o persona con esta necesidad), los cuales están almacenados en grandes colecciones de datos (generalmente almacenada en computadoras) (Manning et al., 2009). Dependiendo del sistema, no necesariamente computacional, en el cual estemos realizando nuestra consulta, el conjunto de objetos en estas colecciones podrían ser reemplazados por documentos, imágenes, vídeos, sonidos, sitios web, entre otros. Hoy en día, la recuperación eficiente en enormes cantidades de datos y de fuentes heterogéneas (varias modalidades), sigue siendo un gran reto para los investigadores dedicados a este proceso (Campos et al., 2014), (Markov, 2014), (Strotgen, 2015), (Kopliku et al., 2014).

Por otro lado, el Aprendizaje Multi-modal involucra información relacionada de múltiples fuentes (modalidades). Por ejemplo, los datos de audio y visuales para el reconocimiento de voz tienen correlaciones en un "nivel medio", como fonemas y visemas (poses de labios y movimientos); es difícil relacionar *pixels* a las ondas de audio o espectrogramas (Ngiam et al., 2011). Cada modalidad se caracteriza por poseer dife-

[1] http://www.internetlivestats.com/twitter-statistics/
[2] https://zephoria.com/top-15-valuable-facebook-statistics/

rentes propiedades estadísticas, lo cual hace difícil ignorar el hecho de que ellas vienen de diferentes canales de entrada (Srivastava y Salakhutdinov, 2014).

En una configuración multi-modal, los datos consisten de múltiples modalidades de entrada, donde cada modalidad tiene un tipo diferente de representación y estructura correlacional. Por ejemplo, los documentos (texto) son usualmente representados como unos vectores de conteo de palabras discretos, mientras que una imagen es representada usando intensidades de sus *pixels* o las salidas de algunos extractores de características, los cuales poseen valores reales.

Si juntamos los conceptos de IR y Aprendizaje Multi-modal se obtiene un nuevo paradigma de búsqueda que permite la recuperación de información desde diferentes modalidades, esto se denomina Recuperación de Información Multi-modal. Por ejemplo, un usuario puede simplemente buscar por un documento o una clase de documentos y el sistema de recuperación de información multi-modal respondería con un conjunto de documentos, imágenes o cualquier objeto de otra modalidad, como respuesta a su consulta. Esta investigación se enfocará principalmente en la recuperación de información dentro y entre dos modalidades, texto e imágenes.

En un sistema de recuperación multi-modal se pueden realizar las siguientes búsquedas (Wang et al., 2014):

- **Búsqueda *Intra-modal*:** Dentro de una misma modalidad. Fue extensamente estudiado y ampliamente utilizado en diversas investigaciones (Deng et al., 2012), (Hutchinson et al., 2013), (Huang et al., 2013), (Shen et al., 2014). Algunos ejemplos son: la recuperación de documentos web y la recuperación de imágenes basado en contenido.
- **Búsqueda *Cross-modal*:** Habilita al usuario explorar los recursos mas relevantes de diferentes modalidades. Por ejemplo: un usuario puede utilizar un *tweet* para recuperar fotos y vídeos relevantes de otra fuente de datos o buscar una relevante descripción textual.

La IR dentro de una misma modalidad (*Intra-modal*) no es un tema trivial y el problema se complica aún mas si trabajamos con varias modalidades (*Cross-modal*). Para resolver este problema, se necesita de un conjunto de funciones que puedan mapear varias modalidades dentro de un mismo espacio semántico o desarrollar una función que pueda establecer una relación entre cada uno de los espacios semánticos de las diferentes modalidades.

En la actualidad se lograron importantes avances en las técnicas de Aprendizaje de Máquina o *Machine Learning* (ML). Uno de los más exitosos avances en este campo fue la creación de modelos y algoritmos conjuntamente conocidos como Aprendizaje Profundo o *Deep Learning* (DL). Esta área de investigación incluye una familia de algoritmos de ML que logran modelar abstracciones de alto nivel en los datos, empleando arquitecturas profundas compuestas de múltiples transformaciones no lineales Wan et al. (2014). El principal objetivo de estos algoritmos es aprender funciones complejas

que dirijan los datos de entrada hacia una salida en particular sin utilizar características manuales, conocimiento *a-priori* o conocimiento del dominio (Wan et al., 2014).

Los algoritmos mas conocidos de DL son las Redes Neuronales Profundas o *Deep Neural Network* (DNN) y fueron exitosamente aplicadas en la ejecución de diversas tareas de aprendizaje, tanto en el enfoque supervisado como en el no supervisado, siendo en este último en donde se destacaron por encima de otros modelos. Estos algoritmos, por lo general, se componen de varias capas de unidades lógicas llamadas "neuronas", cada una de las cuales reciben un conjunto de entradas de las neuronas de la capa anterior, le aplica alguna función de activación no lineal y como resultado obtiene un valor de salida que es enviado a las neuronas de la capa posterior, con el objetivo de aprender una representación semántica de los datos de entrada. La idea fundamental es que cada neurona aprenda una representación diferente de los datos y que al final toda la DNN pueda extraer las características mas importantes de los mismos u otras interpretaciones que se quiera obtener de la red. Debido a sus sobresalientes resultados, lograron posicionarse como el estado del arte en diversas áreas de investigación (Krizhevsky et al., 2012), (Deng y Yu, 2014) (Wang et al., 2016), (Szegedy et al., 2015) (Salakhutdinov y Hinton, 2009).

En esta investigación se utilizarán algunos modelos supervisados y no supervisados de DNN en el desarrollo de un modelo general que pueda establecer una relación entre dos espacios semánticos, texto e imágenes, con la finalidad de poder recuperar información entre estas dos modalidades. Específicamente, se desarrollaron tres modelos, dos de los cuales se entrenan de manera independiente para poder recuperar información de forma *Intra-modal* dentro de cada espacio semántico. Y un tercer modelo que permita establecer una relación *Cross-modal* entre ambos espacios.

Modelo en el espacio de las imágenes: primeramente, se utiliza una DNN muy conocida en el campo denominada Red Neuronal Convolutiva o *Convolutional Neural Network* (CNN) (modelo de aprendizaje supervisado), esta red posee dos partes importantes: un conjunto de capas convolutivas, las cuales se encargan de la extracción de las características mas importantes de las imágenes y un conjunto de neuronas totalmente conectadas para poder intercambiar información semántica entre los resultados de las capas convolutivas. Usualmente esta red se utiliza en el proceso de clasificación de imágenes. Otro de los modelos que se utiliza es el *Deep Autoencoder* (DAE) (modelo de aprendizaje no supervisado) el cual tiene a su vez también dos partes: el codificador o *encoder*, el cual tiene como objetivo transformar los valores de entrada en un vector de mayor o menor dimensionalidad aplicando funciones de activación no lineales a través de las capas de neuronas, y el decodificador o *decoder*, cuya función es reconstruir los valores de entrada aplicando también funciones no lineales. La intuición del DAE es la de reconstruir los datos de entrada utilizando varias capas de neuronas, las cuales realizan la tarea de codificación para aumentar o reducir la dimensionalidad, mientras se mantiene información semántica de los datos. Esta investigación se enfoca en la correcta extracción de las características utilizando una CNN para posteriormente reducir la dimensionalidad de estas con un DAE, con la finalidad de obtener un vector representativo de baja dimensión y con información semántica de las imágenes de entrada que agilice el proceso de recuperación de imágenes.

Modelo en el espacio de los textos: el modelo para el tratamiento de los textos es muy similar al de las imágenes, extraer características principales y reducir su dimensionalidad, pero en el caso de los textos primeramente se utiliza una forma de representación de los textos llamado *Paragraph vector*, el cual consiste en representar piezas de texto de longitud variable con una representación de longitud fija. El algoritmo propuesto en (Le y Mikolov, 2014) (modelo de aprendizaje no supervisado) representa cada documento o sentencia por un vector, el cual es entrenado para predecir palabras en un documento. El concepto de generar un vector representativo de un documento esta muy relacionado con otro concepto llamado *Word representation*, que consiste en representar una palabra generalmente como un vector (Turian et al., 2010). Uno de los métodos más usados para representar una palabra es llamado *Word embbedings* el cual utiliza una DNN para poder obtener un vector de números reales representativos de cada palabra. Después de obtener un vector representativo de los documentos o sentencias de texto, se utiliza otro DAE para reducir su dimensionalidad y obtener un vector representativo de baja dimensión de los textos de entrada para el proceso de recuperación de textos.

Al final se evalúa un nuevo modelo que establezca una relación entre los vectores de imágenes y textos con el objetivo de realizar el proceso de recuperación multi-modal.

Este trabajo posee similaridades con (Wang et al., 2016), el cual actualmente posee el estado del arte en la tarea de recuperación de información multi-modal. Sin embargo, existen importantes diferencias: (1) Los modelos empleados en esta investigación tanto para el tratamiento de las imágenes y los textos son diferentes, los cuales se centran en la correcta extracción de las características de cada modalidad y su propia reducción dimensional. (2) Se desarrollará un nuevo modelo *Cross-modal* que tiene el objetivo de establecer una relación entre estas modalidades y no la creación de un espacio semántico común entre ellas. (3) Por último, el modelo general propuesto en esta investigación, utilizará conjuntos de datos reales, compuestos de imágenes y textos sin ningún pre-procesamiento adicional. Con las diferentes pruebas y evaluaciones realizadas en el modelo de recuperación de información multi-modal propuesto, se obtuvieron resultados que son comparables con otros enfoques e investigaciones orientadas a la misma tarea.

1.1. Planteamiento del Problema

La representación de las características y la reducción dimensional de los datos, son procesos muy importantes en el rendimiento de los modelos de recuperación de información multi-modal. Los actuales modelos de DL utilizados para realizar esta tarea, se enfocan principalmente en solucionar sólo uno de estos procesos. Debido a la gran cantidad de datos heterogéneos que existen, se necesita de un modelo que pueda realizar ambas tareas, extraer las características mas predominantes de los datos y reducir la dimensionalidad de los mismos, con la finalidad de mejorar el rendimiento de la tarea de recuperar información de diversas modalidades (en el caso de esta investigación se utilizarán texto e imágenes).

1.2. Objetivos

Demostrar que un modelo híbrido compuesto por modelos supervisados y no supervisados de redes neuronales profundas, pueden equiparar el rendimiento de los actuales métodos de recuperación de información multi-modal al utilizar información real.

1.2.1. Objetivos Específicos

1. Investigar y analizar el funcionamiento de las redes neuronales convolutivas en los procesos de extracción de características y clasificación de imágenes.
2. Evaluar y comparar el rendimiento de las Máquinas de Boltzmann Restringidas y de los *Autoencoders* en el proceso de pre-entrenamiento de los *Deep Autoencoders* en términos de error en la reconstrucción.
3. Evaluar el método *Paragraph vector* para la generación de los vectores representativos de los textos de entrada.
4. Desarrollar y evaluar el rendimiento del modelo *Intra-modal* de Recuperación de Imágenes, denominado *CNN-DAE*.
5. Desarrollar y evaluar el rendimiento del modelo *Intra-modal* de Recuperación de Textos, denominado *D2V-DAE*.
6. Desarrollar y evaluar un modelo *Cross-modal* con la finalidad de establecer una relación entre ambas modalidades.
7. Evaluar el rendimiento del Modelo General propuesto con los resultados obtenidos en el estudio del estado del arte.

1.3. Contribución de la Tesis

Esta investigación aporta:

- Modelos de recuperación de información *Intra-modal* para textos e imágenes, los cuales están compuestos de dos de los procesos más importantes en este tarea: (1) extracción de las características mas predominantes de los datos de entrada, y (2) la reducción dimensional de los vectores representativos. Estos dos procesos se encuentran inmersos dentro de cada modelo *Intra-modal* propuesto.
- Un nuevo modelo de recuperación de información multi-modal el cual utiliza principalmente datos reales, esto quiere decir que los datos de entrada son considerados crudos o *raw data*, sin ningún tipo de pre-procesamiento previo.

- Una nueva forma de utilización del conjunto de datos COCO, el cual es comúnmente utilizado en la tareas de detección de objetos y en la generación de sentencias naturales que describen una imagen. En esta investigación se procesará de una forma especial este conjunto de datos, con la finalidad de que pueda ser utilizado en la evaluación del modelo de recuperación de información multi-modal propuesto.

1.4. Organización de la Tesis

El Capítulo 2 brindará algunos conceptos básicos y avanzados que se utilizarán en el desarrollo del modelo general. En el Capítulo 3 se describirán investigaciones del estado del arte relacionadas al tópico propuesto. Los modelos propuestos en este trabajo se describirán a detalle en el Capítulo 4. Los resultados obtenidos con todos los conjuntos de datos se muestran en el Capítulo 5. Al final se brindará las conclusiones y trabajos futuros de la investigación en el Capítulo 6.

Capítulo 2

Marco Teórico

Esta investigación principalmente envuelve conceptos generales de Recuperación de Información (IR) y Aprendizaje Profundo (DL). En el presente capítulo se describirán estos conceptos y los métodos de extracción de características y reducción dimensional a ser usados en el desarrollo del modelo de recuperación de información multi-modal propuesto.

2.1. Recuperación de Información (*IR*)

En (Manning et al., 2009) la recuperación de información se define como:

"IR es encontrar un material (generalmente documentos) de naturaleza no estructurada (generalmente texto), que satisface una necesidad de información, desde dentro de grandes colecciones de datos (generalmente almacenada en computadoras)".

También se indica que desde sus inicios hasta la actualidad, el proceso de IR está siendo usado por cientos de millones de personas todos los días, por ejemplo cuando ellos usan un buscador (*Google chrome*, *Mozilla firefox*, entre otros) o solo revisan su correo electrónico.

Según (Manning et al., 2009) existen dos grandes tipos de datos:

- Datos no Estructurados: se refiere a datos que no son muy claros, semánticamente abiertos (fácilmente no se puede entender su significado).
- Datos Estructurados: un ejemplo sería el de las bases de datos relacionales, las cuales son usualmente usadas por diversas compañías para almacenar inventarios de productos y personal.

Según (Baeza et al., 2011), IR es una amplia área de ciencia de la computación

enfocada principalmente en proveer a los usuarios un fácil acceso a la información de su interés, y la definen como sigue:

"*Recuperación de Información trata con la representación, almacenamiento, organización y acceso a los artículos de información, como pueden ser documentos, páginas web, catálogos on-line, archivos estructurados y no-estructurados y objetos multimedia. La representación y organización de la información debería de proveer a los usuarios con un fácil acceso a la información de su interés*".

2.1.1. Componentes de un Sistema de *IR*

Según (Baeza et al., 2011), desde el punto de vista del usuario, existen dos grandes tareas que se tiene que ejecutar en simultaneo: Búsqueda (*Searching*), que es la tarea de buscar documentos que estén relacionados con la consulta deseada y la Navegación(*Browsing*), que es la tarea de navegar sobre los datos, con el fin de encontrar información necesaria para satisfacer nuestra consulta, en la Figura 2.1 se puede visualizar la interacción entre ellas.

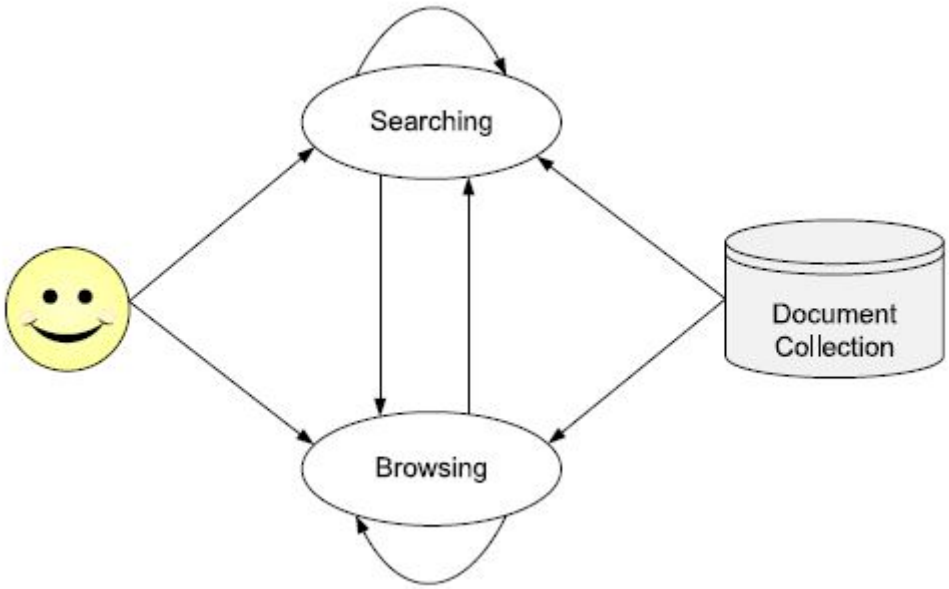


Figura 2.1: Las tareas del usuario (Baeza et al., 2011)

En general y dentro de la interacción del usuario y el computador, el proceso de IR consiste de 3 componentes principales (Kanhabua, 2012): indexación de documentos, procesamiento de la consulta y la recuperación de los documentos. Los componentes se pueden apreciar en la Figura 2.2.

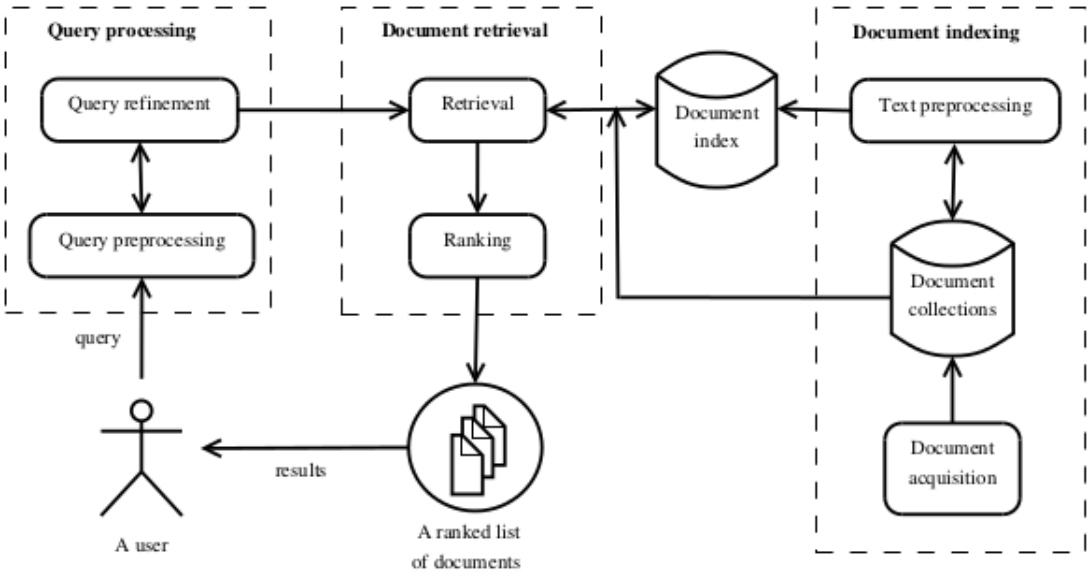


Figura 2.2: Componentes básicos de un sistema de *IR* (Kanhabua, 2012)

2.1.1.1. La Indexación de Documentos

Con el objetivo de que los documentos recuperados, debido a una consulta de un usuario, se retornen lo más rápido posible, los documentos deben de estar indexados. El proceso de transformar los documentos en índices es llamado Indexación de Documentos, el cual posee dos partes:

- La adquisición de documentos: este sub-proceso se refiere a la obtención de los documentos, escaneando libros en documentos digitales o almacenando páginas web.
- El pre-procesamiento del texto: antes de que los documentos sean indexados, el sub-proceso del pre-procesamiento del texto debe de ser realizado. Este sub-proceso puede incluir alguna o todas de las siguientes 5 actividades:

  Segmentación o tokenización: subdivisión del documento en unidades mínimas de análisis.

  Etiquetado: etiquetado de las unidades mínimas de análisis.

  Eliminación de palabras vacías: por ejemplo: de, a, para, entre otras.

  Stemming: conseguir la raíz de la palabra.

  Lematización: es el proceso de llevar una palabra a su forma no conjugada (verbos), no declinada(sustantivos) o a su forma canónica.

2.1.1.2. El Procesamiento de la Consulta

Los componentes principales del procesamiento de la consulta son:

- El pre-procesamiento de la consulta: una consulta debe de ser pre-procesada de la misma forma que el documento, para poder establecer la unión entre los términos de la consulta y los documentos indexados; por ejemplo, una consulta debe de haber pasado por algunas de las cinco actividades mencionadas en el pre-procesamiento del texto, dependiendo del sistema de IR.
- El refinamiento de la consulta: es el proceso de reformular una consulta utilizando términos semánticamente similares, para esto existen dos enfoques:

  Métodos globales: reformular la consulta original, cambiándola con otros términos semánticamente similares. Es independiente de los resultados iniciales.

  Métodos locales: reformular la consulta inicial pero dependiendo de los resultados iniciales, a veces es necesario utilizar un proceso denominado retroalimentación relevante, el cual toma en cuenta la satisfacción del usuario con los resultados recuperados, para realizar un segundo proceso de reformulación de la consulta.

2.1.1.3. Recuperación de Documentos

Es el núcleo del proceso de IR, que a su vez posee varios modelos de recuperación, en los siguientes puntos se explicará cada uno de los de ellos:

- Modelo de recuperación booleano.

  Es el modelo más simple de IR, en donde una consulta q es una combinación de términos y operadores *booleanos* (AND, OR y NOT). Un documento d es modelado como una bolsa de palabras (una lista de términos desordenados), para luego representar la presencia de un término en un documento, usando una medida binaria $\{1, 0\}$ (1 si el término está presente en el documento y 0 en caso contrario). Este modelo ignora el grado de relevancia y solo asume que existen 2 grados, relevante y no relevante, por lo tanto se tiene una función de similitud como la que sigue:

  $$sim(q, d) \in \{1, 0\} \tag{2.1}$$

- Modelo de espacio vectorial.

  En este modelo los documentos son recuperados y clasificados por el grado de relevancia, el cual puede ser medido como la similitud entre una consulta y un documento. Primero, la consulta y los documentos son representados como un vector de pesos de los términos, utilizando un esquema de ponderación de términos. Para lo cual es común utilizar una técnica muy conocida llamada $tf - idf$ (frecuencia del término - frecuencia inversa del documento). El procedimiento

utilizado por esta técnica para la elaboración de los vectores representativos es el siguiente: dado un término w y un documento d, tf es la frecuencia del término w, la cual es normalizada por la frecuencia total de términos en d, la ecuación es la siguiente:

$$tf(w,d) = \frac{freq(w,d)}{\sum_{j=1}^{n_d} freq(w_j,d)} \tag{2.2}$$

Donde, $freq(w,d)$ es la frecuencia del término w en d y n_d es el número del total de términos en d. Por otro lado, idf es la frecuencia inversa del término w en un documento, esto mide la importancia de w con respecto a una colección de documentos. idf puede ser visto como una propiedad discriminativa, donde un término que aparece en muchos documentos es menos discriminativo que un término que aparece en poco documentos. La ecuación es la siguiente:

$$idf(w) = \log \frac{N}{n_w} \tag{2.3}$$

Donde, N es el número total de documentos en una colección y n_w es el número de documentos en los cuales el término w apareció. Finalmente, el valor $tf - idf$ indica la relevancia que tiene un término w dentro de un documento d y puede ser calculado usando la función:

$$tf - idf(w,d) = tf(w,d) \cdot idf(w) \tag{2.4}$$

- Modelo probabilístico.

  En este modelo los documentos son clasificados de acuerdo a la probabilidad de relevancia, la cual tiene que cumplir con dos aseveraciones:

  a) La relevancia es una propiedad binaria, esto indica que un documento puede ser relevante o no relevante.

  b) La relevancia de un documento no depende de otros documentos. Dada una consulta q, R es el conjunto de documentos relevantes y $\bar{R}$ es el conjunto de documentos no relevantes con respecto a q. La semejanza de q y un documento d puede ser calculado como sigue:

$$sim(d,q) = \frac{P(R|d)}{P(\bar{R}|d)} \tag{2.5}$$

- Modelo del lenguaje.

  Originalmente, este modelo fue empleado en el área de reconocimiento del habla para reconocer o generar una secuencia de términos. En años recientes, está siendo utilizado en aplicaciones de IR. El modelo estima una distribución de probabilidad M_D para generar una secuencia de términos en un lenguaje determinado para un conjunto de documentos D. La probabilidad de generar una secuencia de términos puede ser calculada por la probabilidad de generar cada término en la secuencia, la ecuación seria la siguiente:

$$P(w_1,w_2,w_3|M_D) = P(w_1|M_D) \cdot P(w_2|M_D) \cdot P(w_3|M_D) \tag{2.6}$$

2.2. Red Neuronal Artificial (*ANN*)

Una red neuronal artificial es un modelo de aprendizaje de máquina que fue inspirada por el funcionamiento del cerebro biológico (tanto de seres humanos como también de algunos animales). Este modelo fue motivado por dos ideas principales: (1) aprender los principios computacionales del cerebro y duplicar su funcionalidad, y que (2) sería realmente importante entender el comportamiento de cerebro y los principios de la inteligencia humana (Ian Goodfellow y Courville, 2016). En la Figura 2.3 se muestra una neurona biológica, donde las entradas que reciben los estímulos se llaman dendritas y la salida se llama axón. El axón de una neurona biológica, está conectada a las dendritas de otras neuronas para formar una red en el sistema nervioso (Pérez, 2015).

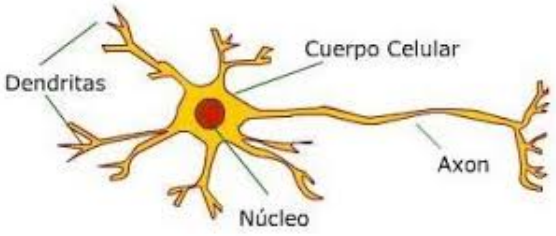


Figura 2.3: Neurona biológica (Pérez, 2015)

La arquitectura general de una *Artificial Neural Network* (ANN) es muy similar a la de una neurona biológica, esta se puede apreciar en la Figura 2.4. Una neurona artificial es una función real de variable real, con varias entradas p y una salida. Cada una de las entradas se multiplica por un coeficiente real W y se suman estos resultados. A los coeficientes W se les llama pesos o *weights*. Por otro lado, se suma un valor real b, llamado coeficiente de desplazamiento, polarización o *bias*. Se puede considerar que se tiene una entrada adicional, que por lo general tiene el valor de 1, el cual se multiplica por el peso del *bias*. Con este conjunto de multiplicaciones y sumas se obtiene el valor n. Finalmente el valor n se conecta a una función real f, que en general es una función no lineal comúnmente llamada función de activación. La salida de las neuronas artificiales se conectan a las entradas de otras neuronas para formar las redes neuronales artificiales. El objetivo de las ANN's es que puedan aprender algunas propiedades o funciones que se necesitan para resolver alguna tarea en específico, por ejemplo: clasificación de datos, regresiones, entre otras.

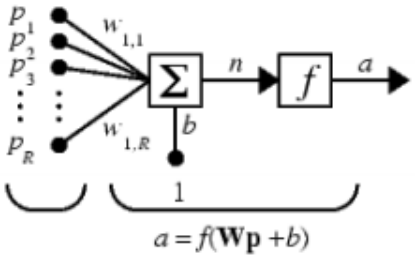
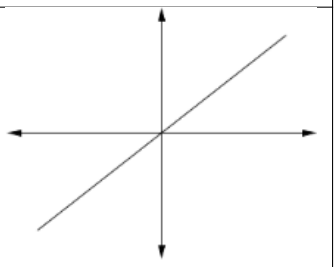


Figura 2.4: Neurona artificial (Pérez, 2015)

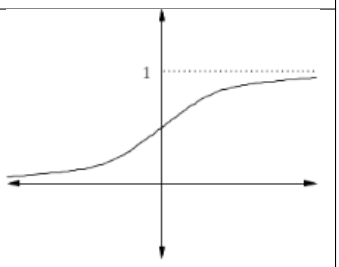
Algunas de las funciones de activación mas conocidas son:

- Función lineal:

Función	Rango	Gráfica
$y = x$	$[-\infty, +\infty]$	

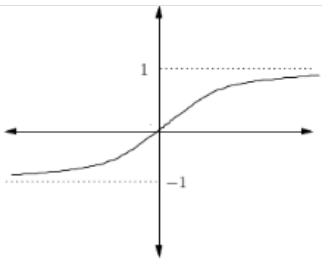
Cuadro 2.1: Función lineal

- Función sigmoidea:

Función	Rango	Gráfica
$y = \frac{1}{1+\exp^{-x}}$	$[0, +1]$	

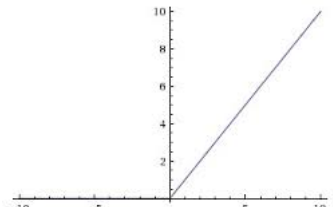
Cuadro 2.2: Función sigmoidea

- Función tangente hiperbólica:

Función	Rango	Gráfica
$y = \frac{\exp^{x} - \exp^{-x}}{\exp^{x} - \exp^{-x}}$	$[-1, +1]$	

Cuadro 2.3: Función tangente hiperbólica

- Función ReLU o *Rectified Linear Unit*:

Función	Rango	Gráfica
$y = max(0, x)$	$[0, max]$	

Cuadro 2.4: Función ReLU

La arquitectura de una ANN mono-capa está compuesta por un conjunto de neuronas ordenadas. Esta arquitectura se puede apreciar en la Figura 2.5

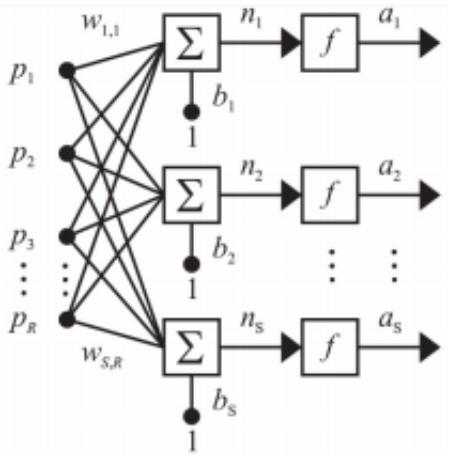


Figura 2.5: Red neuronal artificial mono-capa (Pérez, 2015)

La arquitectura mono-capa permite modelar funciones simples principalmente lineales como por ejemplo las funciones AND y OR (Minsky y Papert, 1969), comúnmente esta arquitectura es llamada *Perceptron*. Con la finalidad de mejorar el rendimiento de una ANN mono-capa, se puede agregar una capa más de procesamiento, teniendo como resultado la arquitectura mostrada en la Figura 2.6, la ANN multi-capa, comúnmente denominada *Perceptron* multi-capa. Esta nueva arquitectura desarrollada también en (Minsky y Papert, 1969), permite modelar funciones complejas. Un ejemplo de esto es la función XOR la cual necesita de al menos dos capas de neuronas y una función de activación binaria (función sigmoidea) para ser modelada. Una red neuronal de múltiples capas esta compuesta por una capa de entrada y otra de salida, las cuales representan la entrada y la salida de toda la red. También una o varias capas ocultas, cuyas entradas provienen de capas anteriores, las salidas se relacionan con capas posteriores.

Una de las grandes ventajas de las ANN's es su capacidad de aproximar funciones. En (Hornik et al., 1989) se manifiesta que el poder real de estos modelos es la capacidad de modelar funciones cuando la no linealidad y la dimensionalidad de un problema en particular se incrementa. En otras palabras, estos modelos pueden manejar funciones

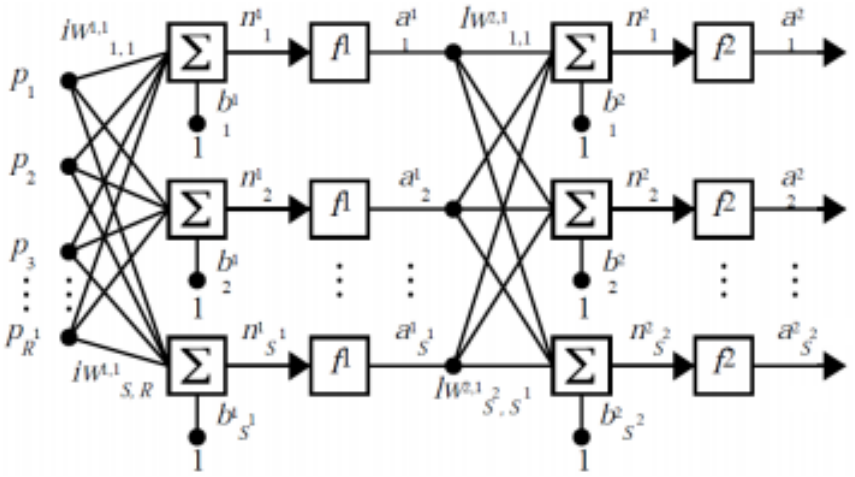


Figura 2.6: Red neuronal artificial multi-capa (Pérez, 2015)

polinómicas de una alta dimensionalidad de una manera muy eficiente.

2.2.1. Aprendizaje

En (Bertona, 2005) manifiesta que durante el funcionamiento de una ANN podemos distinguir claramente dos fases: la fase de aprendizaje o entrenamiento y la fase de operación. Durante la primera fase, la red es entrenada para realizar un determinado tipo de procesamiento. Una vez alcanzado un nivel de entrenamiento adecuado, se pasa a la fase de operación, donde la red es utilizada para llevar a cabo la tarea para la cual fue entrenada.

- Fase de entrenamiento: partiendo de un conjunto de pesos W inicializados aleatoriamente, el proceso de aprendizaje busca un conjunto de pesos que permitan a la red desarrollar correctamente una determinada tarea. Durante el proceso de aprendizaje se va refinando iterativamente la solución hasta alcanzar un nivel de operación suficientemente bueno.
- Fase de operación: una vez finalizada la fase de aprendizaje, la red puede ser utilizada para realizar la tarea para la que fue entrenada. Una de las principales ventajas que posee este modelo es que la red aprende la relación existente entre los datos, adquiriendo la capacidad de generalizar conceptos. De esta manera, una ANN puede tratar con información que no le fue presentada durante de la fase de entrenamiento.

2.2.2. Entrenamiento

La mayoría de los métodos de entrenamiento utilizados en las ANN's consisten en proponer una función de error que mida el rendimiento actual de la red en función de

los pesos. Existen dos funciones de error mayormente utilizadas en el entrenamiento de las ANN's, las cuales brindan el estado en el cual se encuentra la red en un determinado tiempo:

- Suma de errores cuadráticos (SSE): el cual calcula la diferencia que existe entre el valor real y el valor obtenido por la red.

$$SSE = \sum_i (target^i - output^i)^2 \tag{2.7}$$

- Media de los errores cuadráticos (MSE): que es el promedio de la función anterior.

$$MSE = \frac{1}{n} * SSE \tag{2.8}$$

  Donde n es el número de ejemplos de entrenamiento.

El objetivo del método de entrenamiento es encontrar el conjunto de pesos que minimicen (o maximicen) la función. El método de optimización proporciona una regla de actualización de los pesos, que en función de los patrones de entrada, modifique iterativamente los pesos hasta alcanzar el punto óptimo de la red neuronal (Bertona, 2005).

2.2.2.1. Métodos de Gradiente Descendiente

Es el método más utilizado en el entrenamiento de las ANN's, este método define una función $E(W)$ que proporciona el error que comete la red en función del conjunto de pesos W. El objetivo del aprendizaje será encontrar la configuración de pesos que corresponda al mínimo global de la función de error, aunque en muchos casos es suficiente encontrar un mínimo local lo suficientemente bueno (Cauwenberghs, 1993).

El principio general del método es el siguiente: dado un conjunto de pesos $W(0)$ para el instante de tiempo $t = 0$, se calcula la dirección de máxima variación del error. La dirección de máximo crecimiento de la función $E(W)$ en $W(0)$ viene dado por el gradiente $\nabla E(W)$. Luego, se actualizan los pesos siguiendo el sentido contrario al indicado por el gradiente $\nabla E(W)$, dirección que indica el sentido de máximo decrecimiento (De Falco et al., 1998). De este modo se va produciendo un descenso por la superficie de error hasta alcanzar un mínimo local.

$$W(t+1) = W(t) - \alpha \nabla E(W) \tag{2.9}$$

Donde α indica el tamaño del paso tomado en cada iteración, pudiendo ser diferente para cada peso e idealmente debería ser infinitesimal. El tamaño del paso es un factor importante a la hora de diseñar un método de estas características. Si se toma

un paso muy chico el proceso de entrenamiento resulta muy lento, mientras que si el tamaño del paso es muy grande se producen oscilaciones en torno al punto mínimo (Bertona, 2005).

Existe una optimización a este proceso llamado Gradiente Descendiente Estocástico, el cual manifiesta que la actualización de los pesos deben de realizarse para cada paso de un ejemplo de entrenamiento. Este nuevo proceso realiza frecuentes actualizaciones con una alta varianza que hacen que la función objetivo fluctúe fuertemente. Estas actualizaciones e permite saltar a nuevos y potencialmente mejores mínimos locales. En la Figura 2.7 se puede apreciar la diferencia entre los máximos y mínimos locales y globales.

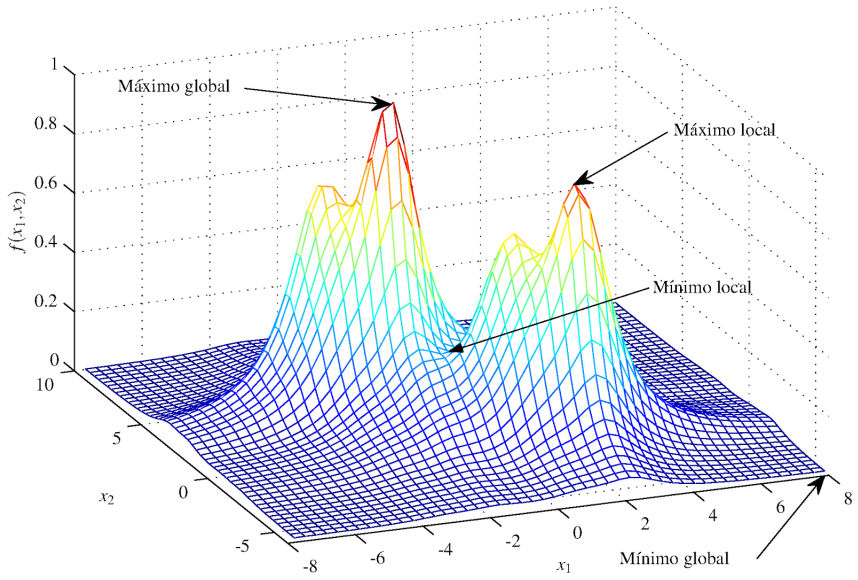


Figura 2.7: Máximos y mínimos locales y globales

2.2.2.2. El Algoritmo de Propagación hacia atrás o *Backpropagation*

Es un algoritmo de entrenamiento de redes neuronales. Su potencia reside en su capacidad de entrenar capas ocultas y de este modo supera las posibilidades restringidas de las redes de una única capa (Olabe, 1998).

Este algoritmo posee dos fases bien identificadas (Olabe, 1998):

- Propagación hacia adelante: esta fase se inicia cuando se presenta un patrón en la capa de entrada de la red. Las unidades de entrada toman el valor de su correspondiente elemento del patrón de entrada y se calcula el valor de activación o nivel de salida de la primera capa. A continuación las demás capas realizarán la fase de propagación hacia adelante que determina el nivel de activación de las otras capas.

- Propagación hacia atrás: una vez se ha completado la fase de propagación hacia adelante se inicia la fase de corrección o fase de propagación hacia atrás. Los cálculos de las modificaciones de todos los pesos en las conexiones empiezan por la capa de salida y continua hacia atrás a través de todas las capas de la red hasta la capa de entrada.

2.2.2.3. Un poco de Historia

Los primeros modelos de ANN's fueron modelos lineales con la finalidad de aprender un conjunto de pesos w_1, w_2, ... , w_n y calcular una respuesta de salida $f(x, w) = x_1 w_1 + ... + w_n w_n$.

Históricamente (Ian Goodfellow y Courville, 2016), existieron 3 grandes épocas en el desarrollo de estos modelos:

- Cibernética (1940-1960): en los inicios de este periodo se desarrollo un modelo lineal el cual podía reconocer dos tipos de categorías de acuerdo si $f(x, w)$ era positivo o negativo, en este modelo los pesos tenían que ser asignados manualmente (McCulloch y Pitts, 1943). En 1950 se desarrollo un modelo llamado *Perceptron* que podía aprender los pesos definiendo las categorías de los datos de entrada (Rosenblatt, 1958). Luego se utilizó este mismo modelo en la creación de *ADALINE* el cual podía predecir un número real a partir de los datos (Widrow et al., 1960). Los modelos antes mencionados fueron llamados Modelos Lineales. Debido a que los Modelos Lineales poseían limitaciones en las funciones que podían representar, la utilización de las ANN's se redujo considerablemente.

- Conexionismo (1980-1990): la idea central es que un largo numero de unidades computacionales simples (neuronas) pueden lograr un comportamiento inteligente cuando son conectadas. Uno de los principales conceptos desarrollados en este periodo fue el de la Representación Distribuida. La intuición es que cada elemento de entrada a un sistema debería ser representado por muchas características, y a su vez, cada característica debería de ser involucrada en la representación de varios elementos de entrada. El mayor logro desarrollado en este periodo fue el exitoso uso del algoritmo *Back-propagation* para entrenar las ANN's (Rumelhart et al., 1988). Los reclamos poco realistas de algunos investigadores y la creación de otros modelos de Aprendizaje de Máquina como: *Kernel Machines* y los Modelos Probabilísticos, hicieron decaer la popularidad de las ANN's hasta el 2007.

- Aprendizaje Profundo (desde 2006): este periodo se explicará en la siguiente sección.

2.3. Aprendizaje Profundo (*DL*)

Se considera al aprendizaje profundo como la evolución de las ANN's, desde el punto de vista de su arquitectura estos nuevos modelos poseen muchas más capas que las ya conocidas redes Multi-Capa, esto les otorga poseer varios niveles de abstracción. El surgimiento de estos modelos se debe a que actualmente se cuenta con gran cantidad de datos que pueden ser utilizados para entrenamiento de modelos más profundos y la mayor capacidad de procesamiento que brindan los nuevos tipos de *hardware*, como las Unidades de Procesamiento Gráfico (GPU), los cuales agilizan el tiempo de procesamiento de estos modelos.

Desde el 2006, el aprendizaje profundo ha emergido como una nueva área de investigación de ML, que logra un gran poder y flexibilidad para aprender representaciones reales. Estas representaciones son tratadas como una jerarquía anidada de conceptos, en donde cada concepto es definido en relación con conceptos más simples. Durante años pasados, las técnicas desarrolladas en las investigaciones de aprendizaje profundo, están impactando grandes trabajos en el procesamiento de señales e información, con enfoques nuevos o tradicionales (Deng y Yu, 2014). En la imagen de la Figura 2.8, se puede ver un diagrama de venn de la ubicación de DL , dentro de los enfoques de *Artificial Intelligence* (AI).

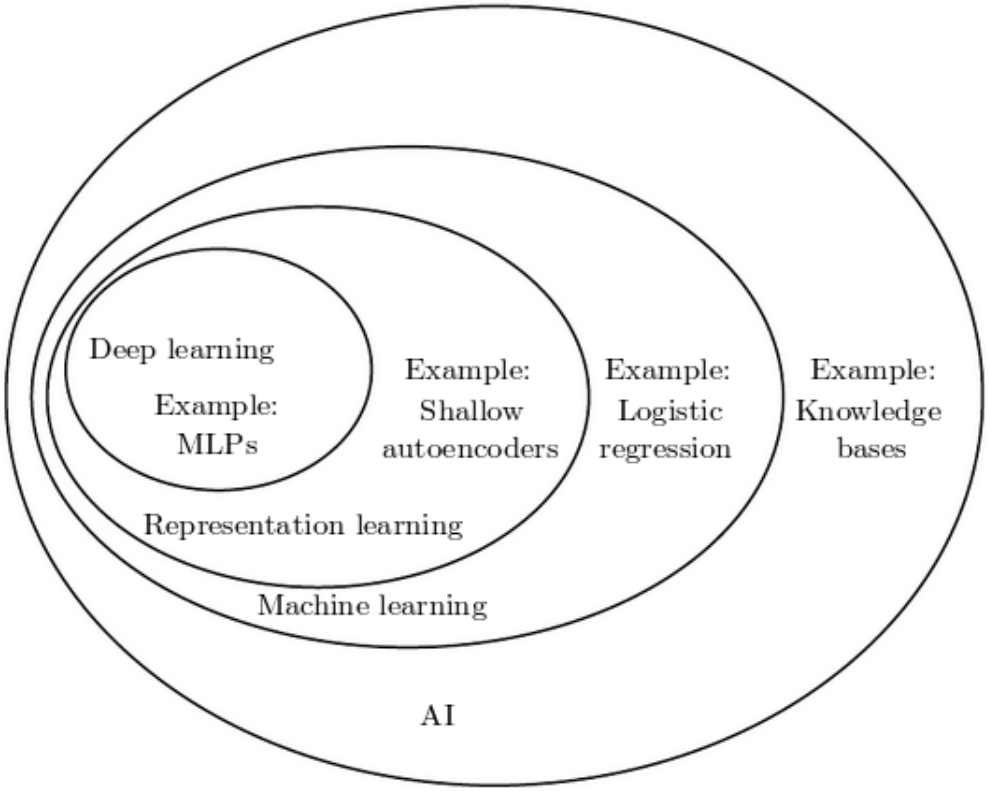


Figura 2.8: Diagrama de venn de *DL* (Ian Goodfellow y Courville, 2016)

Un ejemplo practico del concepto de DL se puede apreciar en la Figura 2.9, en donde se propone el desarrollo de una función de reconocimiento que logre identificar

un objeto a partir de un conjunto de *píxels*, la cual es una tarea muy complicada (Ian Goodfellow y Courville, 2016).

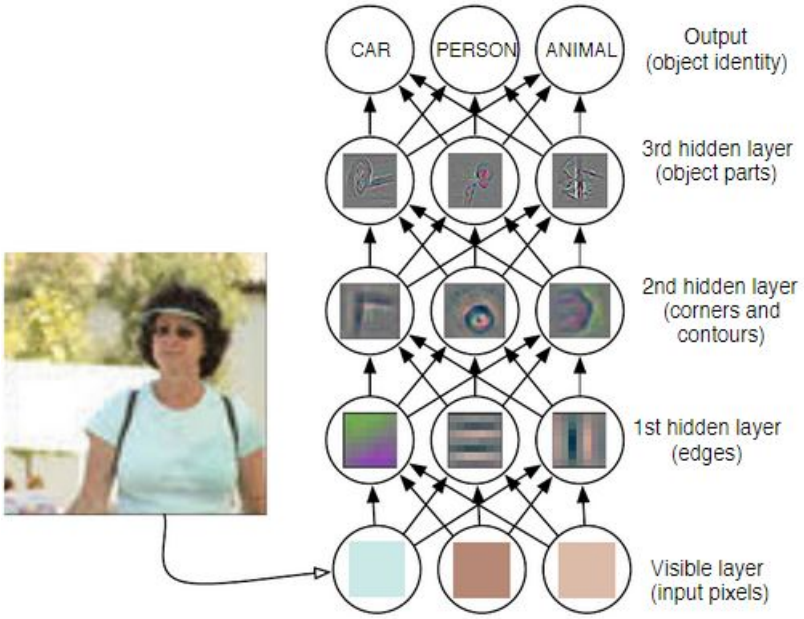


Figura 2.9: Procesamiento de imágenes con *DL* (Ian Goodfellow y Courville, 2016)

DL resuelve esta dificultad dividiendo el reconocimiento completo en un conjunto de series anidadas de reconocimiento parcial, cada una descrita por una diferente capa del modelo. La entrada al modelo es representada como la capa visible, obtiene ese nombre porque contiene las características que fueron observadas en la imagen. Luego el conjunto de capas ocultas extrae características mucho más abstractas de la misma. Son llamadas ocultas porque sus valores no son dados por las entradas del modelo, en vez de eso, este debe determinar cuáles conceptos son útiles para expresar mejor los datos de entrada.

Dados los *píxels* en el procesamiento de imágenes, la primera capa oculta puede fácilmente identificar bordes, comparando la intensidad de los *píxels*. La segunda capa puede buscar por esquinas o contornos extendidos, los cuales son reconocidos como un grupo de bordes. La tercera capa puede detectar partes enteras de un objeto específico, juntando colecciones de contornos o esquinas. Finalmente la descripción de la imagen en términos de partes de objetos, puede ser usada para reconocer los objetos presentes en la misma (Ian Goodfellow y Courville, 2016).

Como se mencionó en el Capítulo 1, esta investigación utilizará tres tipos de modelos de aprendizaje profundo como base para la elaboración del modelo de recuperación de información multi-modal propuesto: (1) Las Redes Neuronales Convolutivas o CNN's como extractores de características sobre las imágenes de entrada, (2) Las Máquinas Restringidas de Boltzmann o *Restricted Boltzmann Machine* (RBM)'s para el pre-entrenamiento de los DAE's, y (3) El funcionamiento de los *Deep Autoencoders* o DAE's como reductores de la dimensionalidad de los vectores representativos de cada una de las modalidades, texto e imágenes. Por lo cual sus definiciones y las principales

características de cada uno de estos modelos serán descritos a continuación.

2.3.1. Redes Neuronales Convolutivas (*CNN*)

Según (Ian Goodfellow y Courville, 2016), las redes convolutivas son redes neuronales que usan una función de convolución en vez de una multiplicación de matrices en al menos una de sus capas. La arquitectura mas común de una CNN esta compuesta por un conjunto de capas convolutivas seguidas de unas capas de neuronas totalmente conectadas, que servirán para realizar el proceso de clasificación, normalmente utilizando una función denominada *Softmax* como función de costo. Uno de los motivos del éxito de este tipo de modelos fue la reducción en los parámetros de entrenamiento del mismo.

2.3.1.1. El Operador Convolución

Es una operación de dos funciones, en la cual una de las funciones se aplica directamente sobre los valores de entrada y la otra función es llamada el *kernel*. Uno de los objetivos de la convolución es la transformación de los valores de entrada con la finalidad de poder extraer o descubrir características o valores mas representativos de los mismos (Ian Goodfellow y Courville, 2016). El modelo de convolución clásico esta dado por la fórmula:

$$s(t) = \int x(a)w(t-a)da \tag{2.10}$$

Esta operación es comúnmente llamada convolución, donde w es llamado el *kernel*. Normalmente el operador convolución es denotado por un asterisco, como se muestra en la siguiente ecuación:

$$s(t) = (x * w)(t) \tag{2.11}$$

Para ejercicios normales los valores de entrada a la función toman valores discretos, para lo cual la fórmula puede ser transformada a:

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \tag{2.12}$$

donde la función x se refiere a la entrada y la función w es el *kernel*. La salida de una convolución es comúnmente denominada como *Feature Map*. El proceso de convolución es comúnmente aplicado en el tratamiento de las imágenes y posee la siguiente función en su aplicación:

$$s(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{2.13}$$

donde I es la imagen y K denota la aplicación del *kernel* sobre la imagen. Debido a que la convolución posee la propiedad conmutativa se puede escribir de la siguiente forma:

$$s(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n). \tag{2.14}$$

En aplicaciones de aprendizaje de máquina y por lo general en el tratamiento de imágenes, antes de la aplicación del *kernel* sobre los datos de entrada, el *kernel* es transformado aplicando una función de rotación (comúnmente una función espejo). Es común confundir la función de convolución con una función denominada *cross-correlation*, la cual para su aplicación el *kernel* no es transformado por ninguna función y es aplicado directamente:

$$s(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n) \tag{2.15}$$

2.3.1.2. Motivación de las *CNN*

Dentro de la aplicación de las CNN's existen 3 ideas importantes que ayudaron a que se posicionarán como el estado del arte en varias tareas:

- Conectividad esparcida: una de las principales diferencias entre una red neuronal totalmente conectada y una convolutiva es que las conexiones de esta última son conexiones locales. En la Figura 2.10 podemos ver un ejemplo de esto. Como las conexiones de la sub-figura de arriba no son totales.

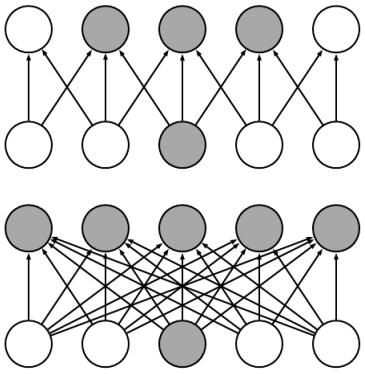


Figura 2.10: Conectividad esparcida (Ian Goodfellow y Courville, 2016)

- Parámetros compartidos: al aplicar un *kernel* sobre toda la imagen, existen conexiones (pesos) que se compartirán y se aplicarán sobre todos los valores de la imagen de entrada. Se puede ver un ejemplo en la Figura 2.11, en la figura las flechas en negrita comparte los mismos pesos.

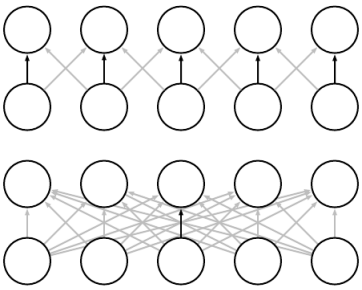


Figura 2.11: Parámetros compartidos (Ian Goodfellow y Courville, 2016)

- Equivariante a la traslación: este concepto significa que si se aplica alguna transformación de traslación a la imagen antes de ingresar a la CNN el resultado serán los mismos pero ordenados de diferente forma. Este concepto se refuerza con la idea de *Pooling*.

2.3.1.3. *Pooling*

El concepto de *pooling* es muy sencillo pero importante, la idea básica es reducir la cantidad de parámetros de la red aplicando una función de reducción de las entradas. Para obtener una intuición de este concepto, imagine que se tiene una imagen de tamaño 32×32, al aplicar *pooling* a la imagen se le reduce su tamaño dependiendo del valor que se requiera. Por ejemplo, si se toma como valor del *pooling* el número 2 el resultado de la aplicación es una imagen de tamaño 16×16, se redujo la imagen a la mitad de su tamaño original.

La forma de reducción de la imagen puede ser realizada de 3 formas según la literatura: *max-pooling* (se escoge el valor mayor), *min-pooling* (se escoge el valor menor) y *mean-pooling* (se escoge el promedio de los valores). Se puede apreciar en la Figura 2.12 la aplicación de *max-pooling*. En la parte de arriba de esa imagen se puede apreciar que se escoge el valor mayor de un pequeño conjunto de neuronas, en la parte de abajo se mueve un poco la ventana de observación hacia la izquierda y también se puede ver que se escoge el valor mayor de cada pequeño conjunto de neuronas. Se puede observar algo muy importante, al mover la ventana de observación los valores anteriormente calculados en la parte de arriba de la figura no se modifican, esto da a entender que al mover la ventana las salidas no se modifican pero que si cambian de posición.

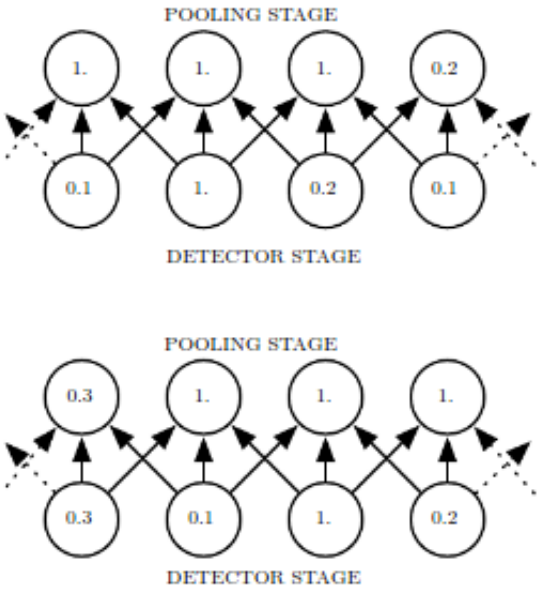


Figura 2.12: *Pooling* (Ian Goodfellow y Courville, 2016)

2.3.1.4. ¿Que es una *CNN*?

Según los conceptos antes explicados una CNN es la aplicación de varias capas convolutivas (una capa convolutiva es un conjunto de *kernels*) y capas de *pooling* seguidas de una red totalmente conectada, con la finalidad de realizar algún tipo de tarea, como: clasificación de imágenes, reconocimiento de objetos, entre otros. Esta arquitectura puede apreciarse en las Figura 2.13 y 2.14

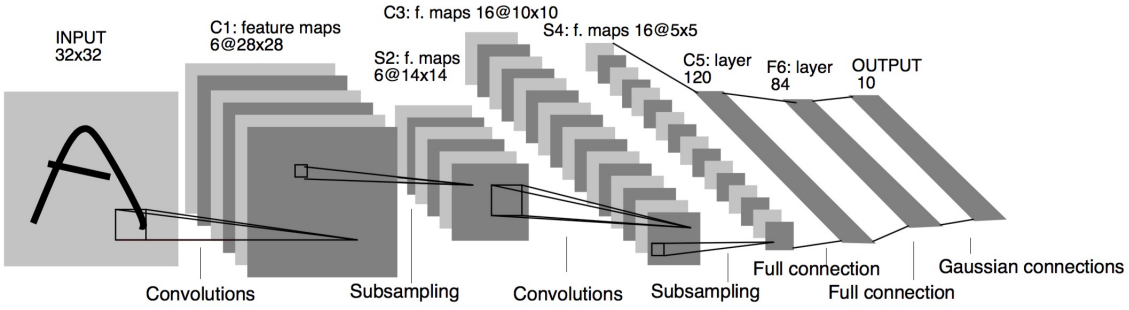


Figura 2.13: *LeNet5* (LeCun et al., 1998)

Debido a que en el desarrollo de esta investigación se necesita de un método de clasificación de imágenes, se tomó como modelo el método de clasificación desarrollado en (Krizhevsky et al., 2012) el cual entrena una CNN profunda con la finalidad de clasificar imágenes de alta resolución, esto como parte de la competición ILSVRC-2012, logrando ganarla por un amplio margen en comparación con otros modelos. Esta competición se desarrolla todos los años y tiene como objetivo comparar el rendimiento de los diversos algoritmos sobre dos tareas en particular: clasificación de imágenes y detección de objetos. Cabe decir que esta investigación se situó como el estado del arte en el año 2012 en las dos tareas antes mencionadas y fue tomada como base en

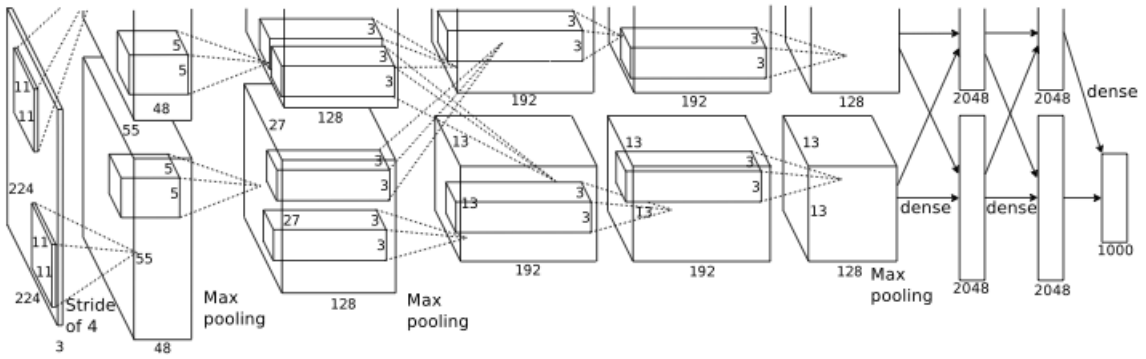


Figura 2.14: Red neuronal convolutiva (Krizhevsky et al., 2012)

muchas investigaciones (Wang et al., 2014), (Wang et al., 2016), (Szegedy et al., 2014), (Szegedy et al., 2015). La arquitectura desarrollada involucró la construcción de una CNN profunda de 5 capas convolutivas seguidas de tres capas totalmente conectadas y al final la función de costo *Softmax* para la etapa de clasificación. Esta función de costo tiene la siguiente expresión:

$$J(W) = -\left[\sum_{i=1}^{m}\sum_{c=1}^{C} 1\{y^{(i)} = c\} log \frac{exp(W^{(c)T}x^{(i)})}{\sum_{j=1}^{C} exp(W^{(j)T}x^{(i)})}\right] \tag{2.16}$$

donde $J(W)$ es la función de costo $Softmax$, m es el número de ejemplos de entrenamiento, C es el numero de categorías, W es la matriz de pesos, x son los datos de entrada, exp es la función exponencial y $1\{\}$ es una función indicador que toma los siguientes valores: $1\ \{un\ valor\ verdadero\} = 1$ y $1\ \{un\ valor\ falso\} = 0$[1].

La parte mas importante de una CNN es el entrenamiento de los *kernels* esto debido que son estos los que extraerán las principales características de los datos de entrada. Para lo cual el entrenamiento primeramente tiene que realizar un paso *Feed-Forward*. Por ejemplo, si se tiene una imagen de tamaño 32×32, y se le aplica 10 *kernels* de tamaño 15×15 se obtiene 10 imágenes (en la literatura a las imágenes de resultado de una convolución se les denomina *Feature Maps*) de tamaño 18×18, se le aplica el proceso de *pooling* de 2×2 y se obtiene una imagen de tamaño 9×9. Si a cada una de las 10 imágenes se le aplica 10 *kernels* de la segunda capa convolutiva con un tamaño de 6×6 cada uno, se obtiene 100 imágenes de tamaño 4×4 y después del correspondiente *pooling* de 2×2, se tiene 100 imágenes de tamaño 2×2 como resultado. Si se junta todas estas imágenes se obtiene un vector con una dimensión de 400, y si las imágenes se encuentran en 3 canales, el vector tendrá una dimensión de 1200. Una vez que se tienen el vector resultante de las capas convolutivas, este se propaga a través de las capas totalmente conectadas.

[1] http://ufldl.stanford.edu/tutorial/supervised/SoftmaxRegression/

2.3.1.5. Proceso de Propagación hacia atrás o *Backpropagation* en las CNN's

Digamos que δ^{l+1} indica el grado de error para la capa $(l+1)$ en la red con la función de costo *Softmax*, como observación W y b es la matriz de pesos y el *bias* en esa capa y (x, y) es el par dato-etiqueta. Si la capa l esta totalmente conectada a la capa $l+1$, entonces el error para la capa l esta dado por [2]:

$$\delta^{(l)} = ((W^{(l)})^T \delta^{(l+1)}) * f'(z^{(l)}) \tag{2.17}$$

donde $f'(z^{(l)})$ es la derivada de la función de activación. El calculo de las gradientes se da por:

$$\nabla_{W^{(l)}} J(W) = \delta^{(l+1)} (a^{(l)})^T \tag{2.18}$$

$$\nabla_{b^{(l)}} J(W) = \delta^{(l+1)} \tag{2.19}$$

donde $a^{(l)}$ es el resultado de la aplicación de la función de activación de la capa anterior. Una vez calculado los errores para cada una de las capas totalmente conectadas se calcula los grados de error al propagar hacia atrás por las capas convolutivas, para lo cual se utiliza la siguiente ecuación:

$$\delta_k^{(l)} = uppooling((W_k^{(l)})^T \delta_k^{(l+1)}) * f'(z_k^{(l)}) \tag{2.20}$$

donde k indica el número del *kernel* y $f'(z_k^{(l)})$ es la derivada de la función de activación. La operacion *uppooling* propaga el error a través de las capas de *pooling*. Finalmente se calcula las gradientes para la actualización de cada *kernel*:

$$\nabla_{W_k^{(l)}} J(W) = \sum_{i=1}^{m} (a_i^{(l)}) * rot90(\delta_k^{(l+1)}, 2) \tag{2.21}$$

$$\nabla_{b_k^{(l)}} J(W) = \sum_{a,b} (\delta_k^{(l+1)})_{a,b} \tag{2.22}$$

Donde $a^{(l)}$ es el valor de entrada a la capa l y $a^{(1)}$ es la imagen de entrada, también representada como x. La operación $(a_i^{(l)}) * rot90(\delta_k^{(l+1)}, 2)$ es la operación de convolución, aplicando una rotación al *kernel*, entre la entrada i y la capa l con el error del *kernel* k. El valor 2 en la ecuación significa que se aplica la rotación por columnas.

[2] http://ufldl.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/

2.3.2. Máquina Restringida de Boltzmann (*RBM*)

Las Máquinas de *Boltzmann* originalmente fueron introducidos con un enfoque conexionista, para aprender distribuciones de probabilidad arbitrarias sobre vectores binarios (Ian Goodfellow y Courville, 2016). Es un modelo basado en energía la cual define una distribución de probabilidad conjunta usando una función de energía:

$$P(x) = \frac{exp(-E(x))}{Z} \tag{2.23}$$

donde $E(x)$ es la función de energía y Z es la función de partición que asegura que $\sum_x P(x) = 1$. La función de energía de una máquina de *boltzmann* esta dada por:

$$E(x) = -x^T W x - b^T x \tag{2.24}$$

donde W es la matriz de pesos y b es el vector de los *bias*.

Las máquinas de *boltzmann* son más poderosas cuando no todas las variables son observadas. En otra palabras trabajan mejor cuando existen variables no observadas o latentes. Teniendo en cuenta esta aseveración, se pudo dividir estas variables en dos subconjuntos, desarrollando una nueva arquitectura denominada Máquina Restringida de *Boltzmann* o RBM. Las RBM's son modelos gráficos probabilísticos las cuales contienen una capa visible y una capa oculta. Este modelo puede ser visto en la Figura 2.15, con lo cual la función de energía cambiaría a la siguiente ecuación:

$$E(v, h) = -b^T v - c^T h - h^T W v \tag{2.25}$$

donde v son los valores de la capa visible, h los valores de la capa oculta, W es la matriz de pesos y b y c son los *bias* correspondientes.[3]. Esta nueva arquitectura se asemeja mucho a la comúnmente conocida *perceptron* multi-capa.

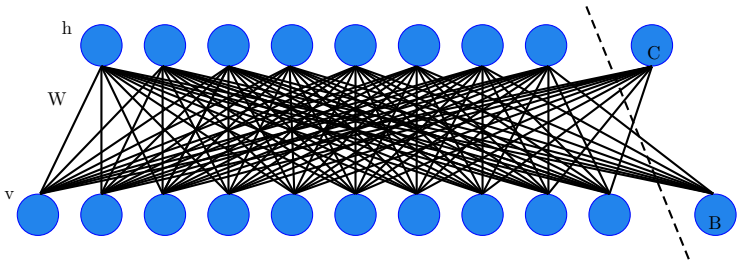


Figura 2.15: *Máquina restringida de Boltzmann*

Debido a la estructura específica de la RBM, las capas visible y oculta son linealmente independientes. Usando esta propiedad se obtiene:

[3]http://deeplearning.net/tutorial/rbm.html

$$p(h|v) = \prod_i p(h_i|v) \tag{2.26}$$

$$p(v|h) = \prod_j p(v_j|h) \tag{2.27}$$

En el caso común, donde v_j y $h_i \in \{0, 1\}$, en otras palabras cuando las entradas sean normalizadas para tener un valor entre 0 y 1, la función de activación de cada neurona está dada por:

$$p(h_i = 1|v) = sigm(c_i + W_i v) \tag{2.28}$$

$$p(v_j = 1|h) = sigm(b_j + W_j^T h) \tag{2.29}$$

donde la probabilidad de que una neurona de la capa oculta (h_i) tenga el valor de 1 dado la capa visible es igual a la función sigmoidea ($sigm$) del *bias* de la capa visible (c_i) más el conjunto de pesos de esa neurona (W_i) multiplicado por los valores de la capa visible(v). La probabilidad de que una neurona de la capa visible (v_j) tenga el valor de 1 dado la capa oculta es igual a la función sigmoidea del *bias* de la capa oculta (c_i) mas el conjunto de pesos de esa neurona (W_j) multiplicado por los valores de la capa oculta(h) (Bengio, 2009).

Debido a este cambio la función de energía obtiene la siguiente forma:

$$F(v) = -b^T v - \sum_i log(1 + e^{(c_i + W_i v)}) \tag{2.30}$$

2.3.3. *Autoencoder* (*AE*) y *Deep Autoencoder* (*DAE*)

El *autoencoder* es un modelo ampliamente utilizado en tareas como aprendizaje de características y clasificación. Puede ser resumida como una ANN de 3 capas, una capa de entrada, una capa oculta o latente y una capa de salida o capa de reconstrucción (Wang et al., 2016). Su función principal es entrenar un conjunto de pesos que puedan reconstruir los datos de entrada; en otras palabras, los datos que son ingresados por la capa de entrada son reconstruidos en la capa de salida por medio de una capa oculta. En los *autoencoders* los datos de entrada ($x_0 \in R^{d_0}$) son codificados en características latentes ($x_1 \in R^{d_1}$) vía una función f_e:

$$x_1 = f_e(x_0) = s_e(W_1^T x_0 + b_1) \tag{2.31}$$

Donde s_e es una función de activación, $W_1 \in R^{d_0 \times d_1}$ es una matriz de pesos y $b_1 \in R^{d_1}$. Luego la característica latente x_1 es de nuevo decodificada en $x_2 \in R^{d_0}$ con otra función f_d:

$$x_2 = f_d(x_1) = s_d(W_2^T x_1 + b_2) \tag{2.32}$$

donde s_d es una función de activación, $W_2 \in R^{d_1 \times d_0}$ es una matriz de pesos y $b_2 \in R^{d_0}$ (Wang et al., 2016). Usualmente se utiliza la función sigmoidea o la tangente hiperbólica como funciones de activación. Todos los parámetros son entrenados con la finalidad de minimizar la diferencia entre los datos de entrada x_0 y su reconstrucción x_2. La distancia euclideana, *Negative Log Likelihood* y *Cross-Entropy* son comúnmente utilizados para medir el error de la reconstrucción. La idea fundamental es que las características latentes conservan alguna semántica de los datos de entrada. Una arquitectura básica de un *autoencoder* se aprecia en la Figura 2.16

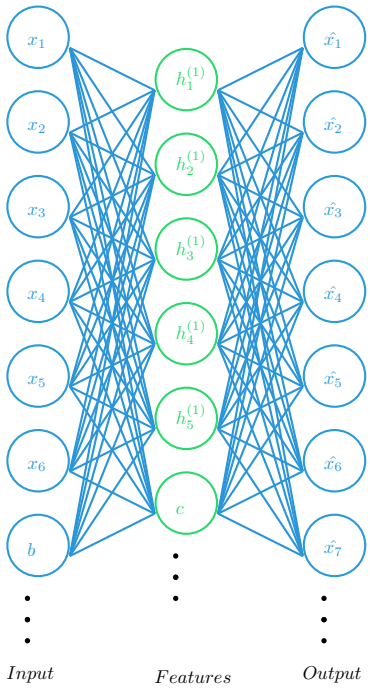


Figura 2.16: *Autoencoder*

La intuición de un *deep autoencoder* es básicamente la misma que un *autoencoder* la diferencia es que el *deep autoencoder* posee mas capas ocultas, ver la Figura 2.17.

Según el desarrollo de las investigaciones anteriores sobre los DAE's, no es recomendable que exista mucha diferencia entre las neuronas de capas consecutivas, por ejemplo si en una capa tenemos 1024 neuronas y en la siguiente capa se tiene 32 neuronas, en algunos casos las gradientes calculadas en el proceso *backpropagation* puede desvanecerse. En algunas pruebas desarrolladas en esta investigación se dieron algunos casos de este inconveniente. Para poder resolver este problema es recomendable que las capas contiguas se diferencien por al menos el doble o la mitad en el número de neuronas, por ejemplo si una capa contiene 1024 neuronas la siguiente debería de tener

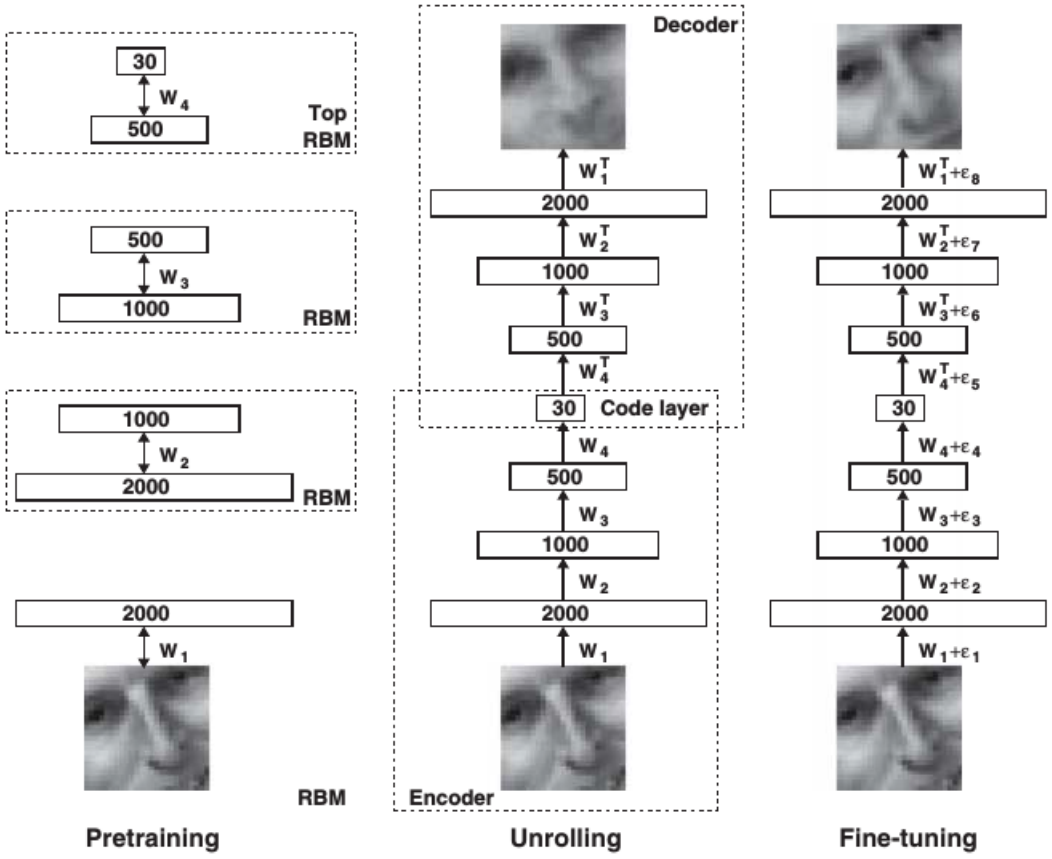


Figura 2.17: Deep autoencoder (Salakhutdinov y Hinton, 2009)

512 neuronas. Por tal motivo es que no podemos utilizar un simple *autoencoder* para reducir la dimensionalidad de los datos de entrada utilizados en esta investigación. Para lo cual utilizamos los *deep autoencoders*.

2.4. Representación de Palabras y Documentos

Muchos de los actuales sistemas y técnicas de procesamiento de lenguaje natural utilizan las palabras como unidades atómicas, donde estas son representadas como índices dentro de un vocabulario. Esta elección se debe a varias razones: simplicidad, robustez y la principal observación de que los resultados obtenidos al entrenar modelos simples con enormes cantidades de datos superan a modelos mucho mas complejos entrenados con menos cantidad de datos. Sin embargo, los modelos simples son limitados en otras tareas de aprendizaje de máquina como: reconocimiento del habla y la traducción automática de textos, en donde al utilizar modelos mucho mas avanzados se obtienen mejores resultados (Mikolov et al., 2013a). Con el avance de las técnicas de ML, fue posible el entrenamiento de modelos más complejos con conjuntos de datos mucho más grandes, obteniendo resultados muchos mejores que los resultados obtenidos con modelos más simples.

Como se mencionó en la Sección 2.2, uno de los mayores avances que se realizaron en ML en los últimos años fueron las técnicas de DL, las cuales están compuestas por redes neuronales artificiales que contienen una o varias capas de procesamiento. Estos modelos son, por lo general, modelos robustos que requieren una gran capacidad de procesamiento y una gran cantidad de datos para obtener buenos resultados. Una de una red neuronal artificial para modelar el lenguaje fue propuesto en (Bengio et al., 2003), donde una red neuronal *feedforward* con una capa de proyección y una sola capa oculta fue usada para aprender un vector representativo de las palabras. Una figura representativa de ese modelo se puede apreciar en la Figura 2.18.

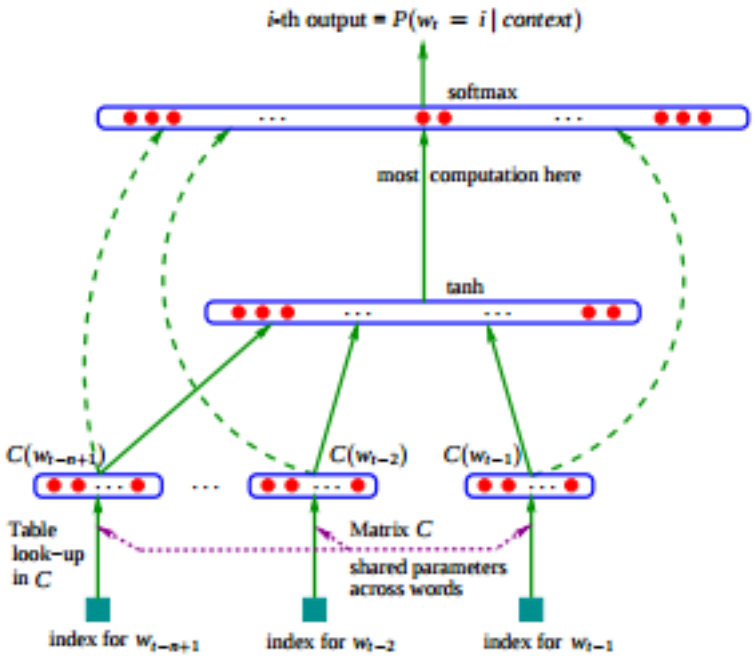


Figura 2.18: Modelo de la red neuronal para modelar el lenguaje (Bengio et al., 2003)

El modelo se entrenó de manera que pueda predecir la siguiente palabra en una oración, por tal razón las entradas al modelo, ubicado en la parte de abajo de la figura, son las palabras anteriores $w_{t-n+1}, ..., w_{t-1}$. Cada una de las palabras anteriores son representados por un vector binario., estos vectores son proyectados sobre la capa de proyección y al final se le aplica una función *softmax* en donde se calcula la probabilidad de que se active la neurona a la que pertenece la siguiente palabra.

En (Mikolov et al., 2009) se modifica el modelo anteriormente mencionado y se reemplazan los vectores binarios de entrada por unos vectores aprendidos con una red neuronal de una sola capa oculta. Estos nuevos vectores son usados para entrenar la red neuronal de modelado de lenguaje.

2.4.1. *Word Embeddings*

En la investigación (Mikolov et al., 2013a) se propone la creación de vectores representativos de palabras con el desarrollo de dos métodos, los cuales tratan de minimizar la complejidad computacional de otras investigaciones desarrolladas con el mismo objetivo, como también mejorar la capacidad de extraer similaridades tanto sintácticas como semánticas de las palabras. Estos dos métodos son llamados *Bag-of-words model* y *Skip-gram model* ver Figura 2.19, el primero de ellos trata de crear el vector representativo de una palabra teniendo en cuenta las palabras que se encuentran alrededor de la misma en un oración o párrafo, en cambio el segundo método trata de crear un espacio semántico en donde se pueda proyectar el vector de un palabra y obtener un conjunto de palabras que guardan una relación de similaridad con la misma a una distancia muy cercana.

Los modelos antes mencionados dominaron el estado del arte en la creación de vectores representativos de palabras en textos. En (Mikolov et al., 2013b) se propuso una mejora sobre estos modelos, básicamente se intentó solucionar el problema de la complejidad computacional. En ese sentido, se utilizaron diferentes algoritmos como la utilización de *Hierarchical softmax* propuesto en (Morin y Bengio, 2005), el cual posee una complejidad menor a la comúnmente utilizada función de costo *Softmax*, entre otras mejoras.

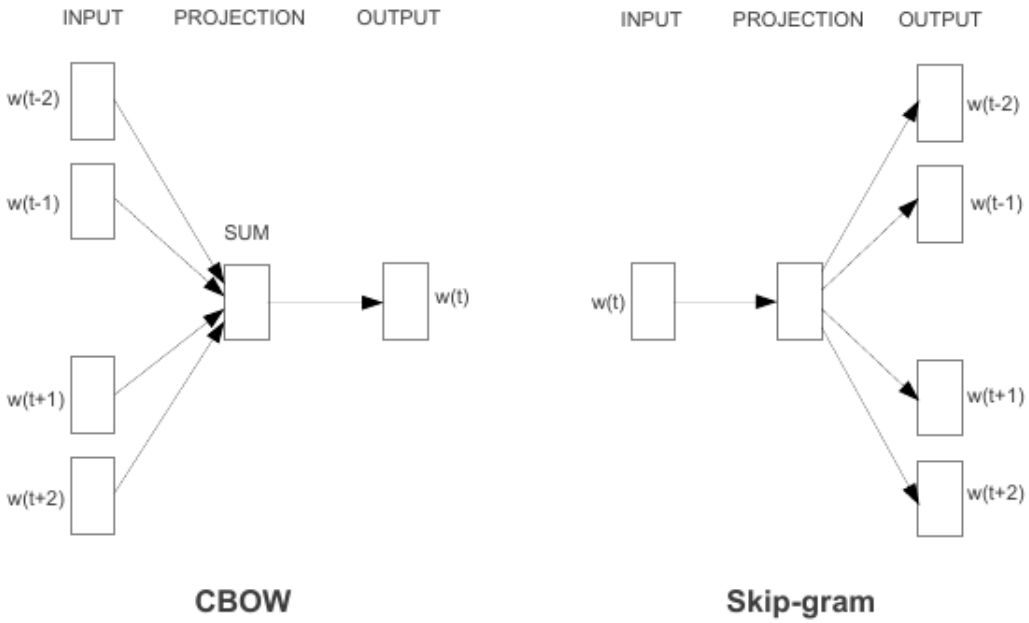


Figura 2.19: Modelos de *word representation* (Mikolov et al., 2013a)

En la Figura 2.20 se puede apreciar una visualización *t-sne*[4] de un conjunto de *word embeddings*, donde palabras similares se encuentran cercanas entre sí.

[4]t-Distributed Stochastic Neighbor Embedding (t-SNE) es una técnica de reducción dimensional que es muy adecuada para la visualización de conjuntos de datos de alta dimensionalidad. Fue desarrollada en (Maaten y Hinton, 2008) y posteriormente fue mejorada en (Van Der Maaten, 2014)

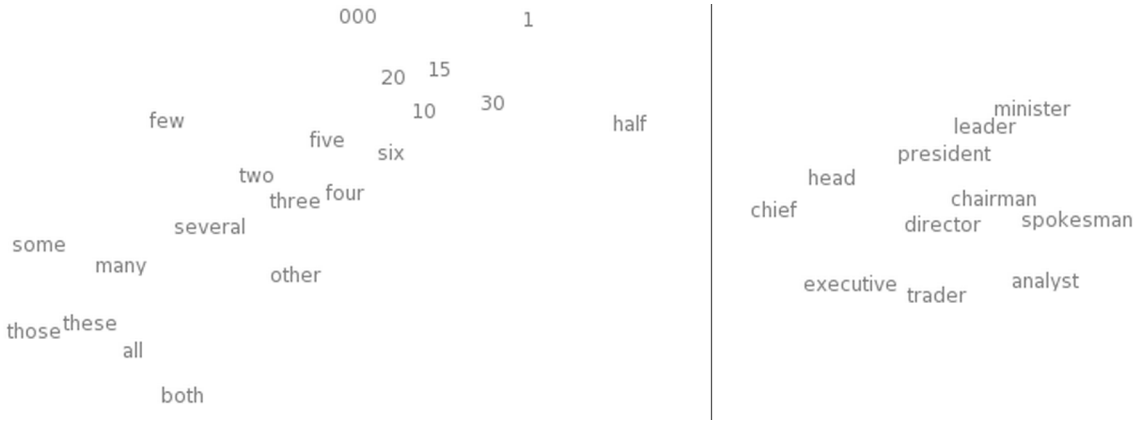


Figura 2.20: Visualización *t-sne* de *word embeddings*

2.4.2. *Paragraph Vector*

El desarrollo de vectores representativos de documentos es una parte muy importante de esta investigación. En (Mikolov et al., 2013b) también se empieza la utilización de los métodos *Bag-of-words model* y *Skip-gram model* para obtener vectores de párrafos de texto. El proceso de obtener vectores de párrafos es similar al de (Mikolov et al., 2013a), el primer paso es obtener los vectores de todas las palabras del *corpus*, luego se crea los ejemplos de entrenamiento o párrafos de texto con las palabras cuyos vectores representativos son muy similares o se encuentran muy cerca en el espacio semántico, y al final con estos párrafos se entrena un nuevo modelo para poder clasificarlos correctamente. Fue una de las primeras investigaciones en tratar de elaborar vectores representativos de párrafos de texto.

En (Le y Mikolov, 2014) se propone la creación de vectores representativos de documentos o párrafos con longitud variable, para lo cual desarrolla el algoritmo *Paragraph vector* que es un modelo de aprendizaje no supervisado. Esta investigación fue inspirada en las dos investigaciones anteriores (Mikolov et al., 2013a) y (Mikolov et al., 2013b), las cuales utilizan ANN's y tratan de predecir una palabra a partir de un texto. La funcionalidad básica de este método es muy parecido al método *Bag-of-words*, en el cual se tiene un conjunto de palabras como entrada al método y se predice la siguiente. La diferencia con el actual es que se le concatena una matriz D al conjunto de palabras de entrada, la cual contiene la representación del *id* del párrafo (vector binario representativo), ver Figura 2.21.

Una vez que este modelo es entrenado, se utiliza el mismo para poder obtener los vectores de los párrafos, en donde se ingresan solo las palabras y conjuntamente con los pesos W y otros parámetros ya entrenados, se obtiene el vector representativo del párrafo. Sobre esta investigación se basa la extracción de características o desarrollo de los vectores representativos de los documentos de texto en la investigación propuesta.

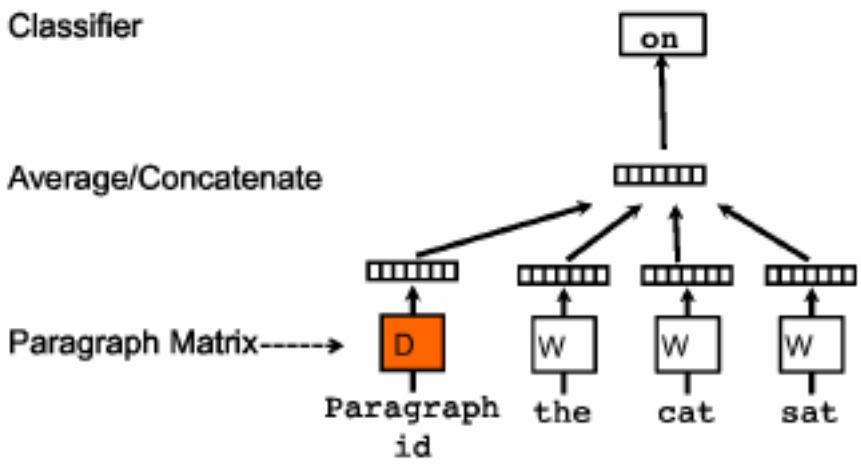


Figura 2.21: Modelo *paragraph vector* (Le y Mikolov, 2014)

En la Figura 2.22 se puede apreciar una demostración de *paragraph vector*, en donde párrafos similares se muestran más cercanos entre sí.

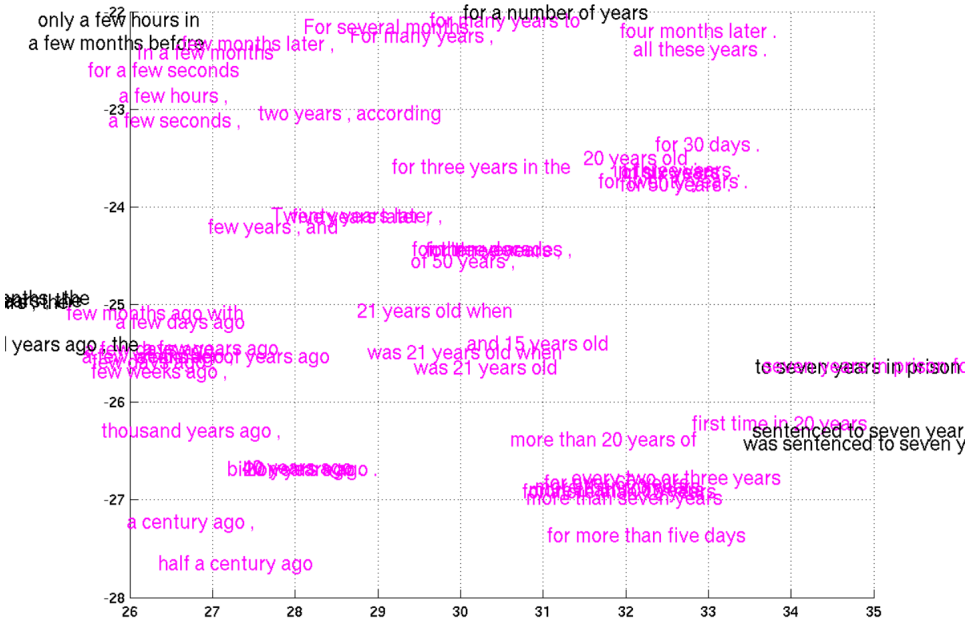


Figura 2.22: Demostración de *paragraph vector* (Cho et al., 2014)

2.5. Conclusiones del Capítulo

En este capítulo se mostraron algunos conceptos y definiciones sobre todo de las dos grandes áreas que contiene esta investigación, como son la tarea de la IR y los modelos de DL. El siguiente capítulo describirá algunas de las investigaciones más importantes en el estado del arte que desarrollen la tarea de recuperación de información multi-modal.

Capítulo 3

Trabajos Relacionados

En este capítulo se explicarán brevemente las principales investigaciones en el estado del arte. Se empezará por investigaciones que abordan la IR multi-modal y que son importantes para poder comparar los resultados de la investigación propuesta en este trabajo. Luego se continuará con algunas investigaciones generales de IR, en donde el uso de modelos de DL lograron excelentes resultados. Y por último se desarrollarán concretamente algunas investigaciones que utilizan modelos profundos en la IR multi-modal. Al final de este capítulo se resumirá cuales son las principales diferencias de nuestro modelo con los del estado del arte.

3.1. Recuperación de Información Multi-Modal

En la investigación (Bronstein et al., 2010) se propone un modelo de aprendizaje supervisado llamado *Cross-modality Metric Learning using Similarity-Sensitive Hashing (CMSSH)*, cuyo objetivo principal es unir datos de dos espacios heterogéneos dentro de un mismo espacio semántico, enfocándose en resolver el problema de aprender una medida de semejanza entre varias modalidades de datos. Para poder resolver este problema utiliza un concepto denominado *Similarity-Sensitive Hashing*, el cual trata de aprender una función de similaridad $\hat{s}$ en un espacio semántico Z a partir de otra función de semejanza s en un espacio semántico R, no específicamente la dimensión del espacio Z sea menor que la de R. La función principal de este nuevo campo Z es la de reducir la distancia de los datos que presenten una mayor semejanza, por lo tanto en este campo se puede posicionar un elemento el cual posee un radio de observación r dentro del cual encontraremos elementos muy similares a este elemento, satisfaciendo la propiedad para que el campo sea sensitivo a la semejanza. Esta investigación amplía el concepto *Similarity-Sensitive Hashing* y lo utiliza sobre datos de dos modalidades diferentes, creando dos diferentes funciones de similaridad: $\xi : X \to H^n$ y $\eta : Y \to H^n$ para lo cual la distancia entre las dos modalidades $d_{H^n}(\xi(x), \eta(y))$ sea pequeña para elementos similares y larga para elementos no similares. Utiliza el algoritmo *AdaBoost* con la finalidad de minimizar la función de costo del modelo general.

En (Shaishav y Raghavendra, 2011) se propone uno de los primeros métodos para aprender funciones *Hash* para cada una de las formas de representación de un objeto o vistas (por ejemplo, una imagen y un texto asociado a un mismo objeto son dos vistas diferentes del objeto), en un conjunto de datos que posee múltiples vistas, llamado *Cross-View Hashing (CVH)*. Las funciones *hash* se utilizarán para resolver el problema de la búsqueda de personas en el idioma Japones, esto se debe a que los directorios de las personas en este idioma pueden ser escritos de diversos formas, por ejemplo *Kanji*, *Katakana*, *Hiragana* y *Romaji*. A parte de solucionar el problema antes mencionado también se desarrollan pruebas con la finalidad de resolver el problema de la búsqueda de personas en diferentes lenguajes. En (Shaishav y Raghavendra, 2011) se sigue un patrón muy parecido a la investigación anterior, pero la diferencia es que traslada las múltiples vistas de un objeto hacia el espacio de una de ellas, con la finalidad de utilizar una función *hash* en este espacio y encontrar elementos similares. Representa cada vista como un código binario compacto llamado *Code Word*. En conclusión la idea principal es desarrollar funciones *hash* que minimicen la distancia *Hamming* de los *code words* de los objetos que son similares sobre cada una de las vistas. Finalmente sus resultados muestran que el proceso de *hashing* propuesto logra un buen rendimiento en la recuperación, comparados con lineas base en los dos problemas tratados.

En (Zhu et al., 2013) se propone un método llamado *Linear Cross-modal Hashing (LCMH)* el cual utiliza las funciones *Hash* con el objetivo de convertir datos de alta dimensionalidad en pequeños vectores binarios, mientras se mantiene las relaciones de cercanía de los datos originales. La idea principal es la de convertir los datos de entrenamiento en k *clusters*, cada uno con un centroide, aplicando un método de *clustering* de tiempo lineal, y luego representar cada ejemplo de entrenamiento como un vector k dimensional el cual posee las distancias a los k centroides. El flujo general del proceso es distribuido en 5 pasos: En el primer paso, se divide los datos de entrada en k *clusters* para cada modalidad; el segundo paso, representa cada ejemplo de entrenamiento con un vector k dimensionalidad; en el tercer paso, se aprende funciones *hash* en un tiempo lineal que preservan las similaridades dentro y entre modalidades; el cuarto paso, se transforma todos los ejemplos de entrenamiento en vectores k dimensionales; y por último, cada uno de estos vectores se convierten en códigos binarios con las funciones *hash* aprendidas anteriormente. Uno de los objetivos de esta investigación es disminuir el tiempo computacional de otras investigaciones relacionadas a la recuperación de información multi-modal y confirmar la escalabilidad y efectividad de su modelo propuesto en comparación con otros modelos.

3.2. Aprendizaje Profundo y Recuperación de Información

En (Salakhutdinov y Hinton, 2009) se muestra cómo la utilización de un modelo profundo logra obtener una mejor representación de los documentos que el método de Análisis Latente Semántico (*LSA*), el cual utiliza la Descomposición de Valores Singulares (*SVD*) para poder reducir la dimensión de los documentos de texto, que

hasta ese momento era el mejor método para esa tarea. Una de las deficiencias del método *LSA* es que es un método lineal y sólo puede capturar correlaciones entre palabras.

También describe un nuevo método de Recuperación de Información llamado *Semantic Hashing*, este nuevo método requiere que la representación de los documentos (el pequeño numero de variables binarias) se coloquen en direcciones de memoria, y que los documentos que son semánticamente similares sean colocados en direcciones de memoria cercanas. Entonces los documentos similares a una consulta pueden ser encontrados a unos pocos *bits* de distancia a la dirección de memoria de la consulta. Uno de sus mayores aportes es su independencia al tamaño del documento.

Si se realiza una descripción de la complejidad algorítmica de los principales modelos, podemos concluir que si se utiliza directamente el enfoque de espacio entre palabras, la complejidad de estos modelos es $O(NV)$, donde N es el tamaño del documento y V es el tamaño del vocabulario, y si se utiliza el enfoque del índice invertido como en $TF - IDF$, la complejidad mejora a $O(BV)$, donde B es el promedio entre todos los términos en el documento de consulta y el número de documentos en los que aparece el término; por ultimo, la complejidad del método *LSA* es $O(VlogN)$. Como conclusión se puede decir que estos últimos modelos dependen del número de documentos y esto los hace demasiado lentos.

Viendo la complejidad algorítmica de los otros algoritmos, el método *Semantic Hashing* produce una pequeña lista de documentos similares a una consulta en un tiempo computacional que es independiente del tamaño de los documentos, para lo cual entrenaron un modelo profundo llamado *Deep Autoencoder*, utilizando un conjunto de RBM's apiladas para realizar el pre-entrenamiento del modelo profundo y luego realizar un proceso llamado *Fine-tuning* sobre todo el modelo con la finalidad de obtener el mejor conjunto de parámetros. Esa investigación logró posicionarse como el estado del arte en la tarea de Recuperación de Información utilizando una DNN en el 2009 (Salakhutdinov y Hinton, 2009).

En (Deng et al., 2012) se propuso una nueva red profunda llamada *Deep Stacking Network* (DSN), que es una arquitectura que agrupa y ordena unas pequeñas ANN's denominadas "módulos", en donde cada módulo contiene solo una capa oculta, la ventaja de esta nueva red es que puede ser altamente paralelizada. Siguiendo ese estudio, en (Deng et al., 2013) se manifiesta que la ventaja de las DSN's, sobre otras redes, es su simplicidad en el aprendizaje. Esta nueva red utiliza la función *Mean Square Error* como función de costo, el cálculo lo realiza entre el valor objetivo y el valor predecido por cada módulo de su arquitectura. La investigación en general es un estudio sobre la mejora en el rendimiento que tiene entrenar un DSN con *Mean Square Error* comparado con otros métodos. Los datos brindados por esta investigación sugieren que las DSN's pueden explotarse aún más.

En (Hutchinson et al., 2013) se desarrolló un nuevo modelo modificando la arquitectura de una DSN, el cual lleva por nombre *Tensor Deep Stacking Network* (TDSN). La idea básica de ese modelo es la de distribuir su procesamiento sobre un conjunto de

procesadores o *clusters*, lo cual mejora su rendimiento, paralelización y escalabilidad.

La construcción de una DNN especializada llamada *Deep Semantic Similarity Model* se publicó en (Huang et al., 2013), la cual es utilizada para clasificar un conjunto de documentos a una consulta en particular. Una proyección no lineal es realizada para mapear una consulta y los documentos en el mismo espacio semántico. En donde la relevancia de cada documento, dada una consulta, dentro de un mismo espacio semántico es calculada por la distancia del coseno. A diferencia de los anteriores métodos, este modelo es optimizado directamente por el *ranking* de los documentos y esto le da un mayor rendimiento. En la investigación también se propone un nuevo método denominado *Word Hashing*, a través del cual, los vectores representativos de las consultas y los documentos, los cuales poseen una alta dimensionalidad, son proyectados en un vector de baja dimensionalidad. La entrada a la DNN propuesta, es un vector de términos de alta dimensionalidad, conteo de términos de una consulta o en un documento sin ninguna forma de normalización y la salida es un vector de términos de baja dimensionalidad en el espacio semántico. Este modelo es utilizado para mapear el vector de términos es su correspondiente vector semántico y para calcular la semejanza del coseno con los vectores semánticos correspondientes. La función de activación utilizada es la tangente hiperbólica. En la búsqueda, los documentos son ordenados dependiendo de sus valores de similaridad semántica.

La investigación anterior fue mejorada en en (Shen et al., 2014) con la utilización de una CNN. Donde se utiliza una secuencia de palabras como valores de entrada, luego se aplica el método *Word Hashing* sobre cada palabra, esta forma de trabajo encuentra solo características locales. Luego de eso se tiene que unir los vectores locales y para esto utiliza el *Max-Pooling*, que es una operación de la CNN, que fuerza a la red a solo retener características locales más representativas. En la última capa se utiliza una capa de neuronas que extrae una representación semántica de alto nivel. La función tangente hiperbólica también es usada como función de activación.

3.3. Aprendizaje Profundo y Recuperación de Información Multi-Modal

En (Wang et al., 2014) se propone un nuevo modelo de recuperación multi-modal llamado *Multi-modal Stacked Auto-encoders (MSAE)*, el cual es construido apilando un conjunto de *Auto-encoders* obteniendo una arquitectura llamada *Stacked Auto-encoders (SAE)* para cada modalidad y proyectando todos los datos dentro de un mismo espacio semántico. Debido a la naturaleza de los propios *Autoencoders* apilados las transformaciones de los datos de entrada, los cuales son no lineales, son mucho mas expresivos que las proyecciones lineales de las investigaciones anteriores. Las principales contribuciones de esta investigación son: (1) se propone un nuevo mecanismo para proyectar vectores de alta dimensionalidad, de cada una de las modalidades, hacia un mismo espacio semántico y de menor dimensionalidad. (2) Se desarrolla una nueva función de costo, la cual toma en consideración semánticas de tipo *intra-modal* y *cross-modal* y

(3) Los experimentos son desarrollados sobre diversos Conjuntos de Datos.

La investigación (Wang et al., 2016) es la continuación de la investigación realizada en (Wang et al., 2014) y contiene actualmente el estado del arte en la IR multi-modal. La idea principal es la de encontrar una función objetivo que capture relaciones semánticas de forma *intra-modal* y *cross-modal* utilizando datos heterogéneos. Para lo cual se proponen dos modelos: (1) modelo que fue desarrollado en (Wang et al., 2014) es un modelo de aprendizaje no supervisado, en el cual se entrenan dos *Stacked Autoencoder* (SAE)'s por separado, uno para trabajar con imágenes y otro con el texto, los modelos se unen mediante la función objetivo propuesta. (2) El segundo es un modelo de aprendizaje supervisado, el cual utiliza dos arquitecturas de redes neuronales muy conocidas: para tratar con las imágenes utilizan una CNN y para el texto utilizan un modelo llamado *Neural Language Model* (*NLM*). Ambos modelos demostraron alcanzar el estado del arte en sus respectivas formas de aprendizaje. Cabe recalcar que utilizan conjuntos de datos con información real.

3.4. Conclusiones del Capítulo

En este capítulo se describieron las más importantes investigaciones en el área de IR multi-modal y DL. Como se describió en las investigaciones principales, el estado del arte en la tarea de IR multi-modal se aprecia en la investigación de (Wang et al., 2016), en el cual se presentan dos modelos, uno de aprendizaje supervisado y otro de no supervisado. La investigación propuesta en el presente trabajo de investigación se diferencia de (Wang et al., 2016) en 3 aspectos fundamentales, los cuales serán desarrollados en el siguiente capítulo.

Capítulo 4

Propuesta

Como se mencionó en el Capítulo 1 el objetivo principal de la investigación es demostrar que un modelo híbrido compuesto por modelos supervisados y no supervisados de redes neuronales profundas, pueden equiparar el rendimiento de los actuales métodos de IR multi-modal al utilizar información real, enfocados específicamente en texto e imágenes. Para la creación del modelo general se utilizará dos modelos de tipo *intra-modal*, los cuales se entrenarán independientemente, uno para el manejo de las imágenes (modelo de recuperación de imágenes) y otro para el manejo del texto (modelo de recuperación de textos). Al final se creará un modelo que pueda relacionar los dos espacios semánticos (texto e imágenes) para poder realizar la recuperación de información de tipo *cross-modal* en ambos espacios (modelo de recuperación multi-modal).

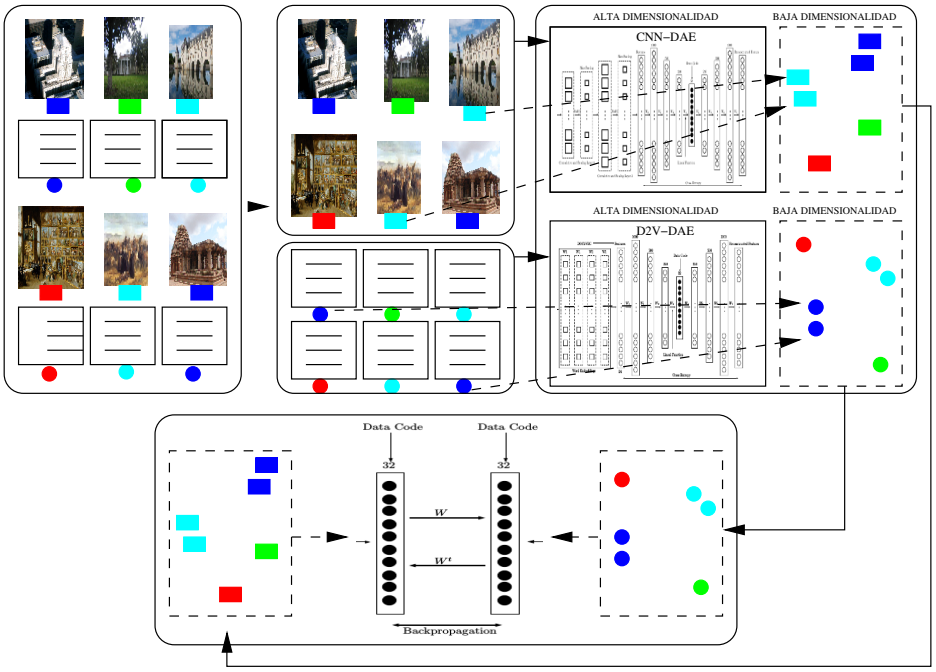


Figura 4.1: Flujo del modelo general

Con fines explicativos, se nombrará al modelo de recuperación de imágenes con

la abreviatura *CNN-DAE* y al modelo de recuperación de textos como *D2V-DAE*. La creación y el funcionamiento de los modelos se detallarán en amplitud en este capítulo. En la Figura 4.1 se puede apreciar el flujo del modelo general propuesto.

4.1. El Modelo de Recuperación *Intra-Modal* de Imágenes *CNN-DAE*

Este modelo contendrá dos procesos generales:

- Extracción de características: para lo cual se entrenará una CNN con todas las imágenes de entrenamiento, con el objetivo de tener un método que pueda extraer las características mas importantes de las imágenes de entrada. Como resultado de este proceso, cada imagen de entrenamiento y de prueba será representada como un vector, el cual contendrá las características más resaltantes de cada una de las imágenes. Este vector por lo general contendrá una alta dimensionalidad. En anteriores investigaciones, la utilización de las CNN's como extractores de características lograron sorprendentes resultados (Vikram, 2015), (Krizhevsky et al., 2012).
- Reducción Dimensional: el vector resultante en el proceso anterior será transformado en otro vector de baja dimensionalidad, utilizando un DAE. El objetivo de este proceso es que cada imagen sea representada por un vector de baja dimensionalidad el cual se llamará *Data Code*; como también, la creación de un espacio semántico *intra-modal* de imágenes. En esta investigación se evaluará tres valores para la dimensión de este vector: 16, 32 y 128.

La ideas fundamentales en la creación de este modelo son las siguientes:

- Primero se entrenará una CNN con todas las imágenes de entrada del conjunto de datos, el entrenamiento se realizará como si fuese una tarea de clasificación, para lo cual se utiliza una función *Softmax* en la capa de salida de la red. Esta parte tiene la finalidad de entrenar los *kernels* de las capas convolutivas de la CNN para que estos puedan extraer las características mas importantes de las imágenes de entrada, como también lograr un alto porcentaje en la clasificación de las imágenes. Como se conoce, este tipo de redes pueden tener diversas configuraciones, como: el numero de capas convolutivas, el numero de *kernels* por capa, el tamaño de los *kernels*, el tamaño del *pooling*, entre los más principales. Cada una de estas configuraciones se probaron en la etapa de prueba de la CNN por cada conjunto de datos y los resultados de las mejores configuraciones se mostrarán en el Capítulo 5. Se puede apreciar en la Figura 4.2 el desarrollo de la CNN la cual se basó fundamentalmente en la investigación de (Krizhevsky et al., 2012).

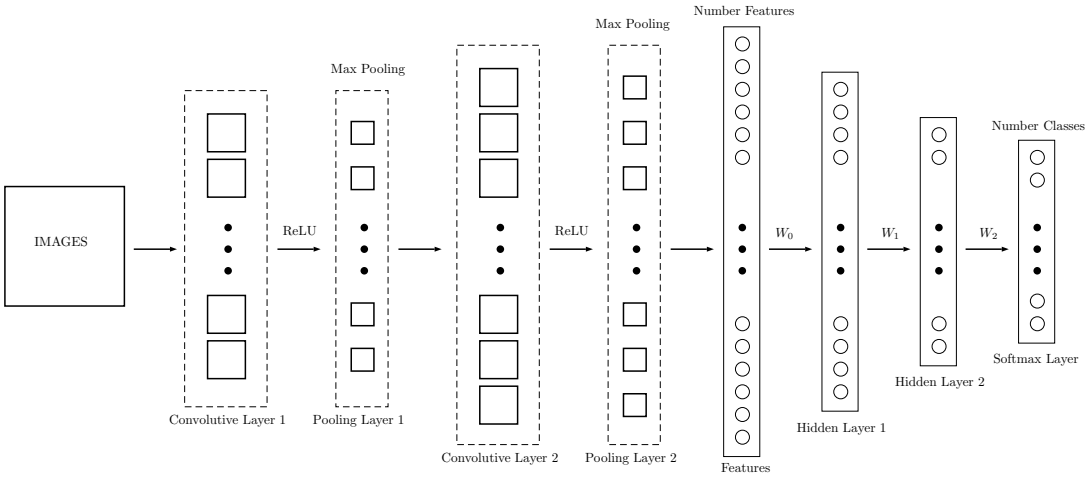


Figura 4.2: Red neuronal convolutiva propuesta

- Una vez que se obtuvieron las mejores capas convolutivas y el mejor conjunto de *kernels* por cada capa que logren un alto porcentaje en la clasificación de las imágenes, lo siguiente es aplicar los *kernels* de las capas convolutivas sobre todas las imágenes de entrada, en otras palabras, se realizará un paso *feed-forward* sobre las capas convolutivas de la CNN entrenada, esto con la finalidad de obtener un vector representativo de las imágenes. El entrenamiento de estas redes se puede apreciar en la Sección 2.3.1.4

- Después de obtener el vector representativo de cada una de las imágenes se entrenará el DAE con todas las imágenes del conjunto de entrenamiento, con la finalidad de reducir la dimensionalidad de este vector. En el ejemplo anterior se obtuvo un vector de 1200 dimensiones, al aplicar el DAE se reducirá a un vector de 16, 32 o 128 dimensiones (*data code*) y se creará un espacio semántico donde se ubiquen estos vectores (espacio de las imágenes). En la Figura 4.3 se puede apreciar el modelo general. Para el desarrollo de los DAE se utilizó como base la investigación en (Salakhutdinov y Hinton, 2009).

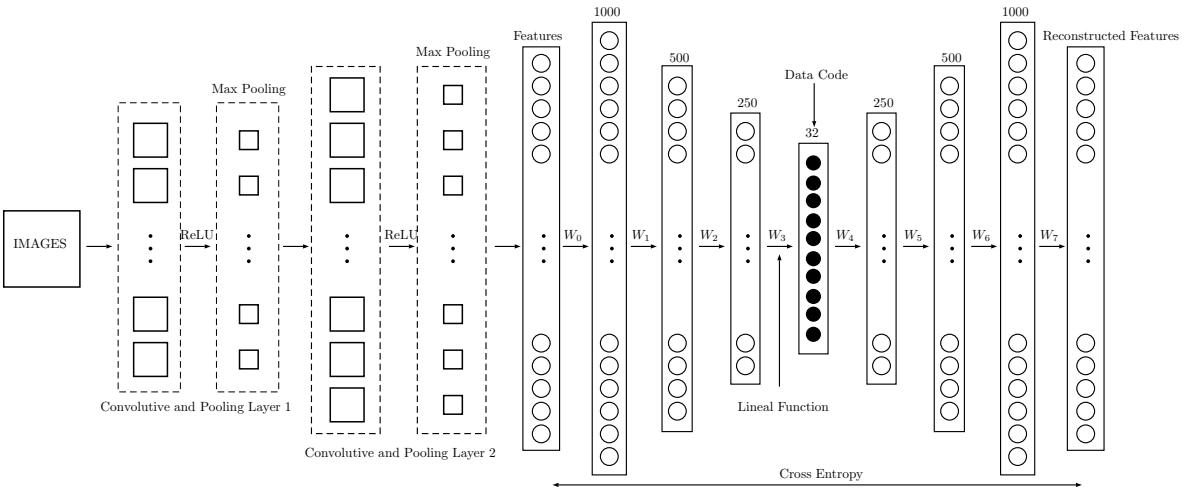


Figura 4.3: Modelo general *CNN-DAE* propuesto

- Para evaluar el rendimiento de este modelo, se calculará lo vectores reducidos de los dos conjuntos, de entrenamiento y prueba, y se obtendrá el grado de *precision*, *recall* y la Media de la Precisión Promedio o MAP.

4.1.1. Aporte del Modelo

Como se describe en el Capítulo 3, en (Wang et al., 2014) se desarrolla un modelo no supervisado de recuperación de imágenes basado en un modelo de DL llamado SAE, este modelo logró posicionarse como el estado del arte en esta tarea. Sin embargo, según las investigaciones realizadas en este modelo, las imágenes de entrada al mismo deben de ser pre-procesadas. Por ejemplo, las imágenes en el conjunto de datos *NUSWIDE* [1] son representadas como vectores binarios. Estos vectores fueron obtenidos aplicando descriptores *SIFT* sobre todas las imágenes. En esta investigación las imágenes de entrada al modelo *CNN-DAE* serán imágenes reales sin ningún tipo de procesamiento previo, ya que está incluido el uso de una CNN como extractor de características de las imágenes.

Debido a que el modelo *CNN-DAE* es propuesto en esta investigación, se incluirá en el Capítulo 5 una sección donde se evaluará su rendimiento.

4.2. El Modelo de Recuperación *Intra-Modal* de Textos *D2V-DAE*

Este modelo también contendrá dos procesos generales:

- Representación de características: para poder representar de forma semántica cada documento o sentencia de texto de entrada, se utilizará el método *paragraph vector* descrito en el Capítulo 2. Este método utiliza un modelo llamado *skip-gram model* con la finalidad de representar un documento o sentencia de texto como un vector de números reales. El objetivo del *skip-gram model* es crear un espacio semántico en donde los vectores de documentos similares se encuentren cercanos entre si.
- Reducción dimensional: para este proceso también se utilizará un DAE como en el caso de las imágenes. El objetivo de este proceso es que cada documento o sentencia de texto sea representado por un vector de baja dimensionalidad el cual también tendrá el nombre de *data code*; como también, la creación de un espacio semántico *intra-Modal* de textos. En esta investigación se evaluará tres valores para la dimensión de este vector: 16, 32 y 128.

Los pasos son los siguientes:

[1] http://lms.comp.nus.edu.sg/research/NUS-WIDE.htm

- Al inicio a cada uno de los documentos en el conjunto de datos se les aplicarán algunos métodos de limpieza del texto, como: eliminación de las palabras vacías, eliminación de caracteres especiales, si los documentos son obtenidos de artículos web se convertirán algunos de los caracteres especiales a su significado original (caracteres *unicode*).
- Luego se utilizará el método *paragraph vector* con la finalidad de entrenar un modelo que pueda crear vectores representativos de los documentos o sentencia de prueba.
- Se calculará los vectores representativos de todos los ejemplos de entrenamiento de cada subconjunto.
- Con los vectores del conjunto de entrenamiento se entrenará un DAE con la finalidad de reducir su dimensionalidad a vectores de 16, 32 o 128 dimensiones (*data code*) y crear un espacio semántico donde se ubiquen estos vectores (espacio de los textos).
- Para terminar, se evaluará el rendimiento de este modelo con las mismas métricas del modelo anterior y el correspondiente conjunto de prueba. En la Figura 4.4 se puede apreciar la arquitectura de este modelo.

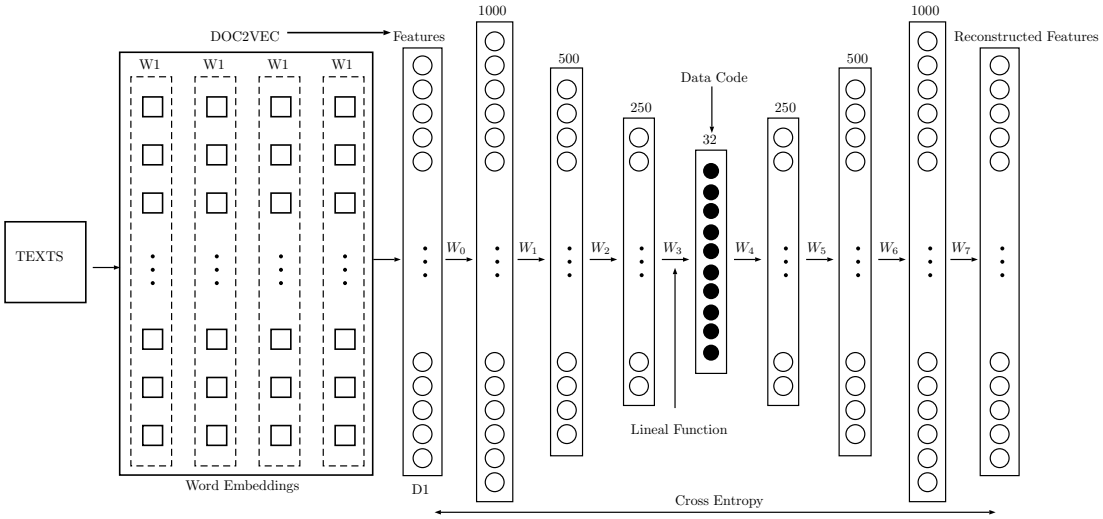


Figura 4.4: Modelo general *D2V-DAE* propuesto

4.2.1. Aporte del Modelo

Este modelo, también como en el caso del modelo *CNN-DAE*, no necesita que los datos sean previamente procesados como en (Wang et al., 2014), dado que se utilizará el método *paragraph vector* para representar los documentos o sentencias de texto.

El modelo propuesto *D2V-DAE* también tiene un aporte principal, y es que las entradas al modelo son documentos o sentencias de texto y no etiquetas como en la mayoría de las investigaciones.

4.3. Modelo de Recuperación Multi-Modal

Una vez obtenido los *data code* de todos los ejemplos de entrenamiento y de prueba tanto de las imágenes como de los documentos o sentencias de texto, el siguiente paso es construir un modelo que pueda enlazar ambos espacios semánticos. El modelo propuesto esta basado en el aprendizaje de un conjunto de pesos de una pequeña red neuronal, como se puede apreciar en la Figura 4.5.

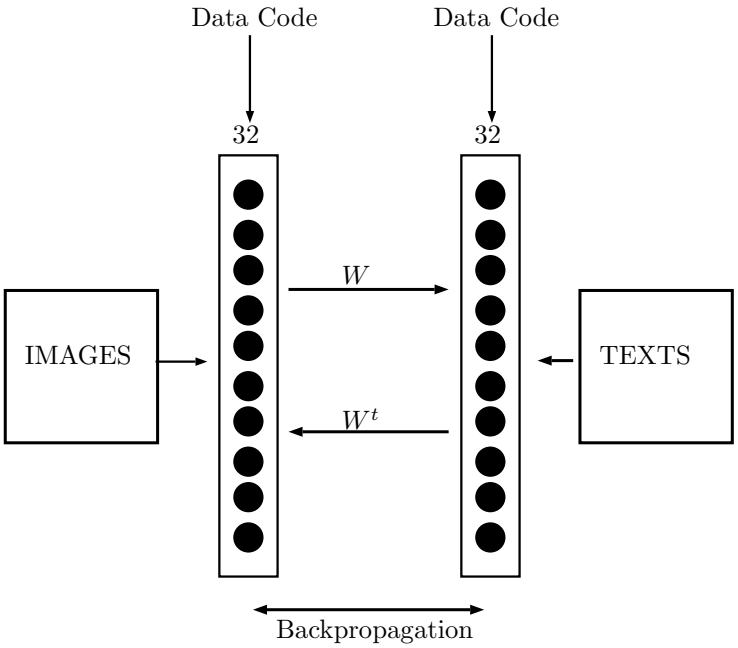


Figura 4.5: Union de modelos propuestos

La idea principal es ingresar todos los *data codes* del conjunto de imágenes y después de un paso *feed-forward* (de izquierda a derecha) compararlos contra sus correspondientes *data codes* de los documentos de texto, de esta manera se entrenará la matriz W. Después de entrenar con todos los *data codes* de las imágenes, se realizará otro paso *feed-forward* (de derecha a izquierda) con los *data codes* de los documentos de texto y compararlos contra sus correspondientes imágenes. Este proceso pulirá los pesos W^T. Al final se obtendrá un conjunto de pesos que relacionará los dos espacios semánticos. Por ejemplo, al ingresar un *data code* de una imagen se obtendrá un *data code* en el espacio de los textos, con estas dos representaciones ya se podrá recuperar tanto imágenes como documentos o sentencias de texto y viceversa. Para medir la similitud de los *data codes* en ambos espacios se utilizará la distancia euclidiana.

4.3.1. Aporte del Modelo

A diferencia de (Wang et al., 2016) y (Wang et al., 2014) en donde tratan de crear un espacio semántico común entre las dos modalidades, la investigación propuesta trata de establecer una relación entre estos espacios. Por lo cual, para poder realizar el proceso

de recuperación de información multi-modal, solo se requerirá de un conjunto de pesos $W \in R^{dxd}$ donde d es el tamaño del *data code*. Debido a que el tamaño de los *data codes* es reducido, el entrenamiento de este modelo tendrá un tiempo computacional bajo.

4.4. Conclusiones del Capítulo

En este capítulo se logró describir la propuesta del trabajo de investigación, teniendo en cuenta algunos detalles en la elaboración de los modelos: (1) para la extracción de características en las imágenes se utiliza una CNN, que es un modelo de aprendizaje supervisado y luego para la reducción dimensional se aplicará un DAE que es un modelo de aprendizaje no supervisado. Por otro lado, para la extracción de características en los textos se utiliza el algoritmo *paragraph vector* que es un modelo no supervisado y para la reducción dimensional se aplicará otro DAE. Debido a la utilización de modelos supervisados y no supervisados, el modelo general propuesto posee un enfoque híbrido. (2)Para el pre-entrenamiento de los DAE's se utilizará las RBM's en vez de los *AutoEncoder* (AE)'s, el motivo de esta decisión se validará en el capítulo 5.(3) Y por último, el modelo general fue diseñado para trabajar con documentos de texto o párrafos que describen las imágenes y no con etiquetas como la mayoría de las investigaciones.

Los modelos mostrados se tienen que ver como unos modelos genéricos, esto debido a que en el Capítulo 5 se detallarán las configuraciones utilizadas dependiendo de cada conjunto de datos. En otras palabras la propuesta mostrada puede cambiar debido a cada conjunto de datos.

Capítulo 5

Pruebas y Resultados

En el presente capítulo se presentarán los siguientes puntos:

- Conjuntos de datos: se describirán las características principales de los conjuntos de datos a utilizar en el desarrollo de las pruebas y evaluaciones.
- Proceso de implementación: esta sección está relacionada con los pasos en la construcción de los modelos, los cuales fueron descritos en el Capítulo 4 donde se elabora la propuesta del trabajo de investigación. Se brindan algunas anotaciones especiales a tomar en cuenta en el proceso de implementación de los modelos *CNN-DAE*, *D2V-CNN* y el modelo de recuperación multi-modal propuesto.
- Las métricas de evaluación: se describen las métricas que se utilizarán en la evaluación de los modelos propuestos.
- ResultadosgGenerales: los resultados generales serán divididos como sigue:
  - Evaluación en el pre-entrenamiento de los DAE's: se evaluará el rendimiento de las máquinas restringidas de Boltzmann (RBM's) y los *Autoencoders* (AE's) al ser utilizados para pre-entrenar los DAE's utilizados en los modelos propuestos.
  - Evaluación del modelo *CNN-DAE*: se evaluará el rendimiento del modelo propuesto *CNN-DAE* comparado con el rendimiento de un *Deep Autoencoder* en el proceso de recuperación de imágenes. Esta prueba se llevó a cabo con la finalidad de tener un punto de referencia al comparar el modelo propuesto en esta investigación contra el desarrollado en (Wang et al., 2014) para el tratamiento de las imágenes. Esta evaluación será realizada con los conjuntos de datos COIL20, MNIST y CIFAR10.
  - Evaluación del modelo general de recuperación multi-modal: por último, se evaluará el rendimiento del modelo general de recuperación de información multi-modal. La evaluación será realizada con los conjuntos de datos WIKI y COCO.

5.1. Conjuntos de Datos

Los conjuntos de datos a utilizar en el desarrollo de la investigación serán divididos en dos partes. Se utilizarán 3 conjuntos de datos para medir el rendimiento del modelo *CNN-DAE*.

- COIL20[1]: imágenes de diversos objetos a escala de grises de 128×128 *pixels*, dividido en 20 categorías cada una de las cuales contiene 72 imágenes. Para su utilización se disminuyó el tamaño de estas imágenes de 128×128 a 32×32 en donde el 70 % de imágenes aleatorias por clase se utilizaron en el conjunto de entrenamiento y el 30 % en el conjunto de prueba.
- MNIST [2]: imágenes de números escritos a mano a escala de grises de 28×28 *pixels*, dividido en 10 categorías. Se utilizó 60000 imágenes en el conjunto de entrenamiento y 10000 en el conjunto de prueba.
- CIFAR10 [3]: imágenes naturales a color de 28×28 *pixels*, dividido en 10 categorías cada una de las cuales contiene 5000 imágenes. Se utilizó 50000 imágenes en el conjunto de entrenamiento y 10000 en el conjunto de prueba.

También se utilizarán 2 conjuntos de datos para medir el rendimiento del modelo de IR multi-modal general, estos conjuntos poseen tanto documentos o párrafos de texto como también imágenes. Estos conjuntos se seleccionaron debido especialmente a los documentos o sentencias de texto que poseen, cabe recalcar que en esta investigación se utilizarán documentos o sentencias de texto como entradas al modelo *D2V-DAE* y no etiquetas, como la mayoría de las investigaciones enfocadas a la misma tarea. Los conjuntos de datos usados para esta parte de los experimentos son:

- WIKI [4]: contiene 2866 pares de documentos e imágenes divididos en 10 categorías. Debido a los diversos tamaños de las imágenes a color, se disminuyeron las mismas a un tamaño de 32×32 *pixels*. De todo el conjunto se utilizaron 2173 pares (imágenes y texto) para el conjunto de entrenamiento y 693 pares para el conjunto de prueba.
- COCO [5]: este conjunto de datos no es comúnmente utilizado en la tarea de recuperar información, pero lo utilizamos porque cumple con las características buscadas. Contiene 82081 imágenes en el conjunto de entrenamiento y 40137 en el conjunto de prueba. En el caso de los párrafos, contiene 414113 párrafos u oraciones en el conjunto de entrenamiento y 202654 en el conjunto de prueba. Como se puede observar existen mas párrafos que imágenes, esto se debe a que

[1] http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php
[2] http://yann.lecun.com/exdb/mnist/
[3] https://www.cs.toronto.edu/~kriz/cifar.html
[4] http://www.svcl.ucsd.edu/projects/crossmodal/
[5] http://mscoco.org/

existen en promedio 5 párrafos por imagen. Las imágenes, como en los casos anteriores, también se disminuyeron a un tamaño de 32×32 *pixels*. Las imágenes se encuentran divididas en 80 categorías.

El motivo de la disminución de los tamaños en las imágenes, es por el tiempo computacional en el procesamiento del modelo *CNN-DAE*. Mientras las imágenes de entrada a este modelo tengan una dimensión mayor, mayor será el tiempo de entrenamiento del modelo y mayores recursos computacionales necesitará. Cabe señalar que los experimentos se realizaron sobre una *Desktop* con las siguientes características:

- Memoria RAM: 15.6 Gb
- Procesador: Intel Core i7-6700k CPU @ 4.00 GHz * 8
- Tipo de SO: 64-bits
- Disco Duro: 500 Gb

5.2. Proceso de Implementación

El proceso de implementación está relacionado directamente con pasos desarrollados, por cada modelo, en la propuesta de la investigación mostrada en el Capítulo 4. Cabe mencionar que el desarrollo se realizó en los lenguajes de programación python y C++ utilizando las librerías OpenCV en las implementaciones. A continuación se brindarán algunos detalles en el desarrollo de los modelos propuestos.

- Los *kernels* de la CNN y los pesos de todas las capas totalmente conectadas de los DAE's fueron inicializados teniendo en cuenta una Distribución Normal o Gaussiana. Tomando una media de 0 ($\mu = 0$) y una desviación estándar de 1 ($\sigma = 1$)
- El entrenamiento de la CNN se realizará teniendo en cuenta el modelo propuesto en (Krizhevsky et al., 2012), por lo cual se utilizarán algunos métodos para evitar el sobre ajuste o *overfitting* como: *Data Augmentation*[6] y *Dropout*[7]. Como también métodos para agilizar el aprendizaje, como la utilización de una función de activación denominada *ReLU*.
- En las diversas investigaciones que utilizan las CNN's, se observan principalmente dos formas de aplicar la convolución:

[6]Creación de nuevas imágenes de entrenamiento aplicando alguna transformación a los datos actuales. Por lo general se aplican rotaciones de 45 y -45 grados; como también aplicación de algún tipo de ruido sobre las imágenes.

[7]Cancelación de algunas neuronas de las capas totalmente conectadas de una DNN en cada iteración del modelo.

  - *CONV-SAME*, en la cual se le agrega a la imagen de entrada un borde de 1 *pixel* de tal forma que la imagen resultado después de la convolución tenga el mismo tamaño que la imagen de entrada.
  - *CONV-VALID* en la cual se aplica la convolución directamente y la imagen de resultado posee menos *pixels* que la imagen de entrada.

  En esta investigación se utilizará la forma de aplicación *CONV-VALID* para las capas convolutivas en la CNN, con la finalidad de reducir la cantidad de parámetros de la red.

- Existe también un factor llamado *Stride* que es el valor que avanza el *kernel* de convolución sobre la imagen. Si este valor tiene el valor de 1, el *kernel* avanza sobre la imagen cada 1 *pixel*, si tiene el valor de 2 avanza cada 2 *pixels* y así sucesivamente. En esta investigación se utilizará un *stride* de 1.
- También para las capas de *pooling* se utilizará el tipo *max-pooling*.
- Se utilizará el método Gradiente Descendiente Estocástico. en el entrenamiento de la CNN y los DAE's.
- En el pre-entrenamiento de los DAE se utilizarán RBM's. Esta decisión se evaluará en la primera sección de los resultados.
- Para el proceso de *Fine-tuning* de los DAE's se utilizará la función *minimize.m* propuesto en la investigación (Hinton y Salakhutdinov, 2006). [8]
- Es bueno considerar un método de normalización de las imágenes al ingresar a algún algoritmo de ML, en esta investigación se probaron 3 de los mismos: *Mean Subtraction*, *Feature Standardization* y *PCA/ZCA Whitening* [9]. Estas pruebas se evaluarán en la sección de resultados.
- Para desarrollar el método *Paragraph Vector* sobre el conjunto de documentos o sentencias de entrada, se utilizará la librería *DOC2VEC*[10].

5.3. Métricas en la Evaluación de la Recuperación

Existen dos aspectos importantes en la evaluación de un sistema de IR:

- Eficiencia: se mide en relación al tiempo de respuesta y al espacio utilizado en un sistema.
- Eficacia: se mide de acuerdo a la calidad en la clasificación de los documentos recuperados.

[8] http://learning.eng.cam.ac.uk/carl/code/minimize/minimize.m
[9] http://ufldl.stanford.edu/wiki/index.php/Data_Preprocessing
[10] https://radimrehurek.com/gensim/models/doc2vec.html

Con el objetivo de medir la eficacia de un sistema de IR existen dos métricas comúnmente utilizadas:

- *Precision*: es la fracción de los documentos recuperados que son realmente relevantes.
- *Recall*: es la fracción de documentos relevantes recuperados.

El presente trabajo de investigación solo abordará la medición de la calidad de los resultados en términos de la eficacia del modelo propuesto, utilizando las medidas de evaluación con recuperaciones no clasificadas, propuestas en (Manning et al., 2009). En donde los documentos recuperados son considerados como un conjunto desordenado de documentos. Las decisiones de un sistema pueden ser clasificadas como: verdaderos positivos (TP), falsos positivos (FP), verdaderos negativos (TN) y falsos negativos (FN), como se puede apreciar en la Tabla 5.1:

	Colección de Documentos	
Predicción del Sistema	Relevante	No Relevante
Recuperado	TP	FP
No Recuperado	FN	TN

Cuadro 5.1: Matriz de confusión de un sistema de *IR* (Manning et al., 2009)

En el contexto de IR se puede calcular los valores para: *precision* y *recall*, como sigue:

$$precision\ p = \frac{TP}{TP + FP} = \frac{\#(elementos\ relevantes\ recuperados)}{\#(elementos\ recuperados)} \tag{5.1}$$

$$recall\ r = \frac{TP}{TP + FN} = \frac{\#(elementos\ relevantes\ recuperados)}{\#(elementos\ relevantes)} \tag{5.2}$$

Otra de las métricas utilizadas en esta investigación es el MAP, la cual es una medida de calidad en los sistemas de Recuperación de Información que demostró tener una buena discriminación y estabilidad en sus resultados (Manning et al., 2009). Esta métrica necesita de otra llamada *Average Precision* que es el promedio de los valores en precisión obtenidos para un conjunto K de documentos existentes, después de que cada documento relevante es recuperado. Básicamente el MAP es el valor del *Average Precision* entre el número de consultas realizadas. Si el conjunto de documentos relevantes a una consulta es $q_j \in Q$ es $\{d_1, ..., d_{m_j}\}$ y R_{jk} es el conjunto de los resultados recuperados clasificados desde el inicio hasta que se llega al documento d_k, entonces la función es dada como sigue (Manning et al., 2009):

$$MAP(Q) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{m_j} \sum_{k=1}^{m_j} Precision(R_{jk}). \tag{5.3}$$

Con la finalidad de mejorar el Proceso de Recuperación, tanto en imágenes como en texto, se propuso comparar los resultados al transformar los *data codes* a dos formas de representación, a espacios de memoria *bits* y a valores reales. En las gráficas de *precision* y *recall*, mostradas en cada evaluación, se visualizarán los resultados obtenidos al transformar los *data codes* a cada una de estas formas de representación y luego aplicarles el proceso de recuperación.

5.4. Evaluación en el Pre-entrenamiento de los *DAE*'s

5.4.1. Desarrollo de *RBM's*

Uno de los algoritmos comúnmente utilizado para el pre-entrenamiento de los DAE's son las RBM's. La manera que se utilizó para realizar el entrenamiento de cada una de las RBM utilizadas en esta investigación se muestra en la Figura 5.1, en la cual también se observa que se utilizó el método *Constractive Divergence*$(CD)_1$. Se tomó esta decisión debido a que en la práctica se demostró que trabaja realmente bien (Hinton, 2010).

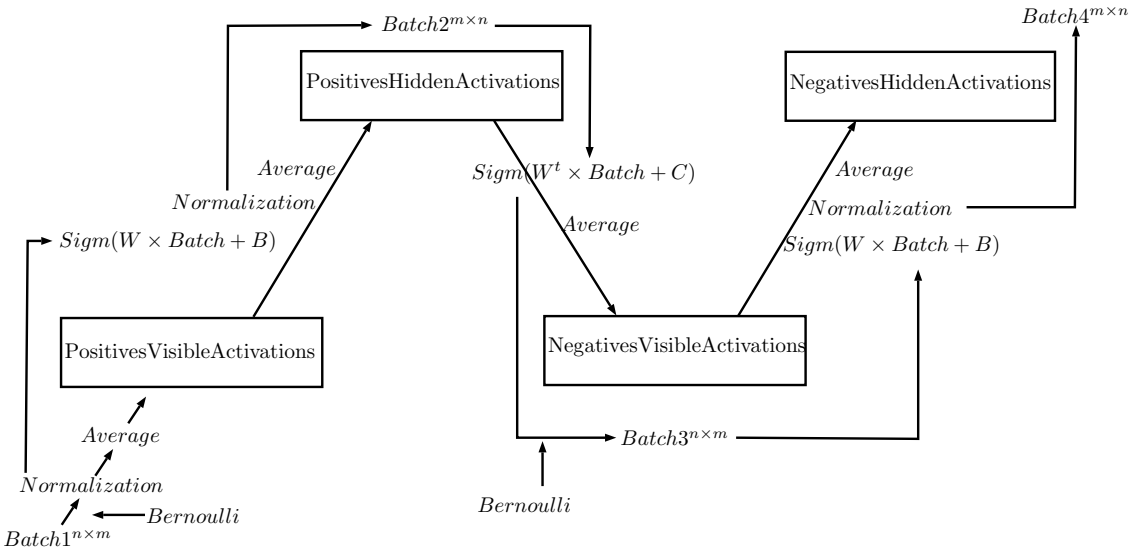
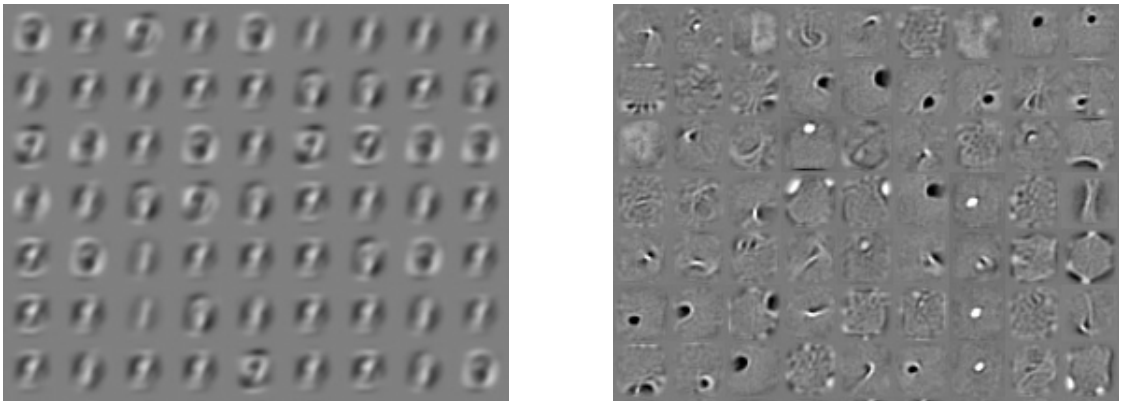


Figura 5.1: Paso para la implementación de una *RBM* con el método CD_1

Uno de los pasos importantes en el entrenamiento de las RBM's es la normalización de los datos después de aplicarles la función de activación. Para lo cual se realizaron pruebas sobre si deben de ser normalizados o no. Cabe mencionar que estos experimentos fueron realizados con el conjunto de datos *MNIST* [11]. En la parte izquierda de la Figura 5.2 se muestran los pesos que se obtuvieron como resultado del entrenamiento de un RBM en el que las salidas de la función de activación fueron normalizados. Se puede apreciar que los pesos de la RBM aprendieron a representar algunos de los datos

[11] http://yann.lecun.com/exdb/mnist/



(a) Normalizado (b) No normalizado

Figura 5.2: Resultados de un *RBM* normalizado y sin normalizar.

de entrada; caso contrario, en el lado derecho vemos que los pesos actúan como *kernels* o extractores de características, algo parecido al concepto de convolución mostrado en el Capítulo 2.

En la figura 5.3 se puede ver como el grado de error va evolucionando a través del entrenamiento en ambos enfoques. Se puede ver que el entrenamiento con un RBM normalizado llega de un grado de error mas alto que el RBM no normalizado.

Según los resultados obtenidos, el enfoque que se utilizarán en el pre-entrenamiento de los DAE's serán los RBM no normalizados, debido a que logran un menor error en la reconstrucción de los datos de entrada y actúan como extractores de características.

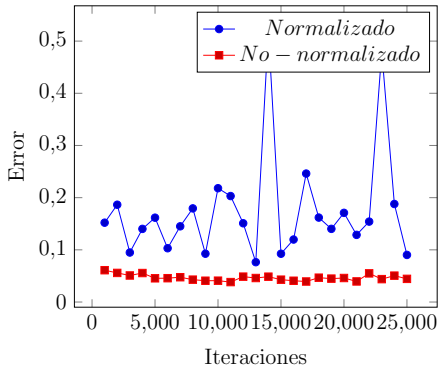


Figura 5.3: Grado de error de un *RBM* normalizado y no normalizado.

5.4.2. Desarrollo de *AE's*

Como parte de las diferentes pruebas en el desarrollo de los modelos propuestos, se tuvo en consideración la utilización de los *Autoencoders*, en el pre-entrenamiento de los pesos del DAE. Por lo cual se entrenó un *Autoencoder* con la finalidad de reconstruir los datos del Conjunto de Datos *MNIST* En la Figura 5.4 se muestran las reconstrucciones obtenidas, ingresando a un AE entrenado un ejemplo de cada tipo de dígito.

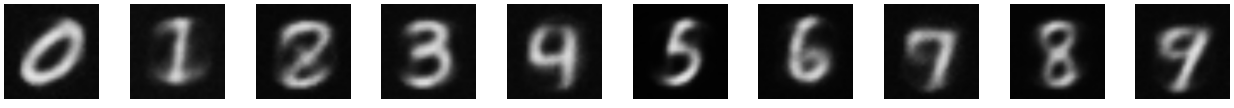


Figura 5.4: Resultados de un *AE*

5.4.3. Comparación entre *RBM's* y *AE's*

En esta parte se comparará los resultados de el pre-entrenamiento de los DAE con RBM's y AE's. Se observa en la Figura 5.5 las reconstrucciones en ambos casos y se puede apreciar que las reconstrucciones cuando se utilizan RBM's son mejores al caso contrario. En la Figura 5.6 se observa también la disminución del error alcanzado por cada uno de ellos. Se puede notar que el grado de error utilizando RBM's es mucho menor al utilizar AE's.

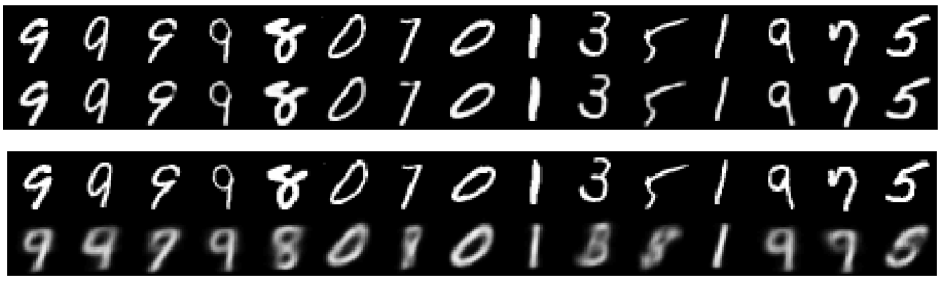


Figura 5.5: Comparación de un *DAE* entrenado con *RBM's* arriba y *AE's* abajo.

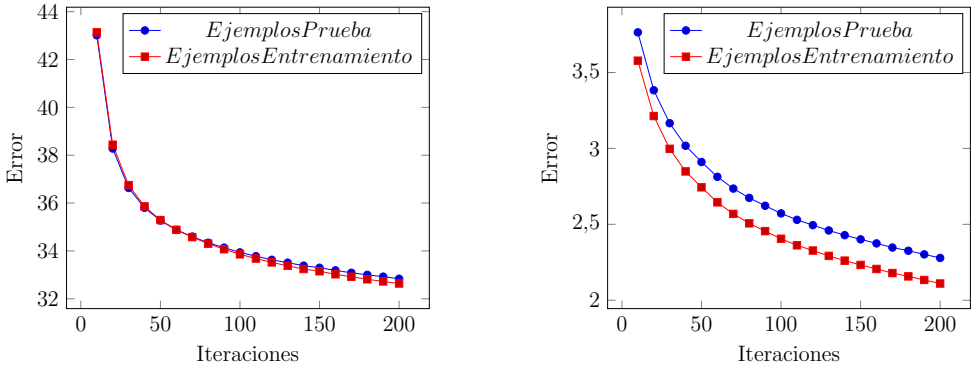


Figura 5.6: Grados de error entre un *DAE* pre-entrenado con *AE's* (izquierda) y un *DAE* pre-entrenado con *RBM's* (derecha).

En esta sección se llegaron a las siguientes conclusiones:

- El grado de error en la reconstrucción de los datos utilizando una RBM no normalizada es mucho menor al utilizar una RBM normalizada.
- Los resultados muestran que en el pre-entrenamiento de un DAE es mucho mejor utilizando RBM's, ya que logran una mejor reconstrucción de los datos.

- Por lo cual, en el pre-entrenamiento de los DAE's utilizados en los modelos propuestos, *CNN-DAE* y *D2V-DAE*, se utilizará las RBM's no normalizadas.

5.5. Evaluación del Modelo *CNN-DAE*

5.5.1. Resultados en COIL20

5.5.1.1. Pruebas con el *DAE*

En la primera fila de la Figura 5.7 se puede apreciar los datos de entrada al DAE y en la segunda fila las reconstrucciones obtenidas. Se observa que las reconstrucciones no se encuentran al 100 % y que contienen algunos errores.

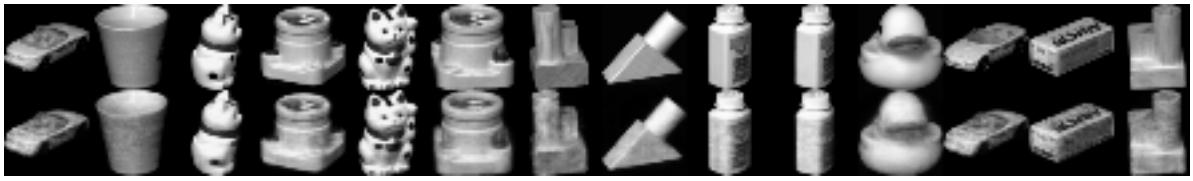


Figura 5.7: Reconstrucción de las imágenes COIL20 con el *DAE*

La Figura 5.8 muestra los resultados luego de aplicar el DAE como reductor dimensional sobre el conjunto de datos COIL20 a un *Data Code* de 32 valores (solo con fines demostrativos). Se pueden observar que se logra una muy buena diversificación en las categorías. Para la visualización de este tipo de resultados se utilizó la librería *T-SNE*[12].

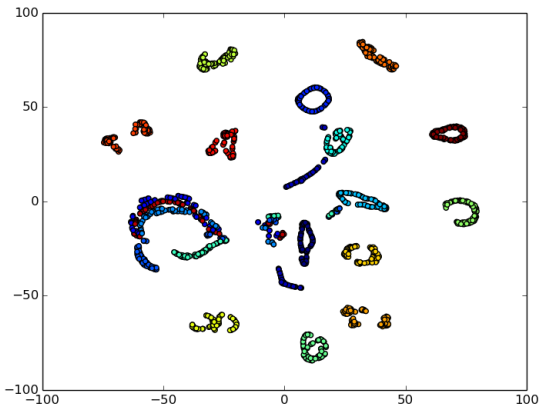


Figura 5.8: Proyección en dos dimensiones de la reducción dimensional en COIL20 con el *DAE*

[12] https://lvdmaaten.github.io/tsne/

La gráfica *precision* y *recall* se muestra en la Figura 5.9 en la cual se compara al transformar los datos a *bits* y a valores reales. Con los resultados de la gráfica anterior se obtiene un valor para el MAP de 0.67. En la Figura 5.10 se observan algunas de las imágenes recuperadas utilizando un DAE, la imagen de arriba a la izquierda es la imagen de prueba y las imágenes restantes son las recuperadas ordenadas por su posición en la recuperación, en el orden: de izquierda a derecha y de arriba hacia abajo.

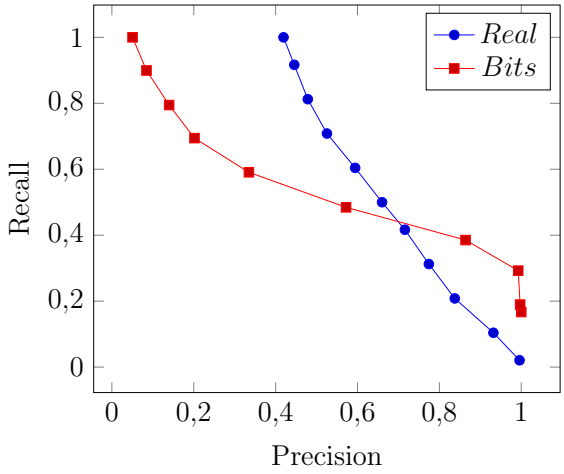


Figura 5.9: Resultados *precision-recall* COIL20 con *DAE*

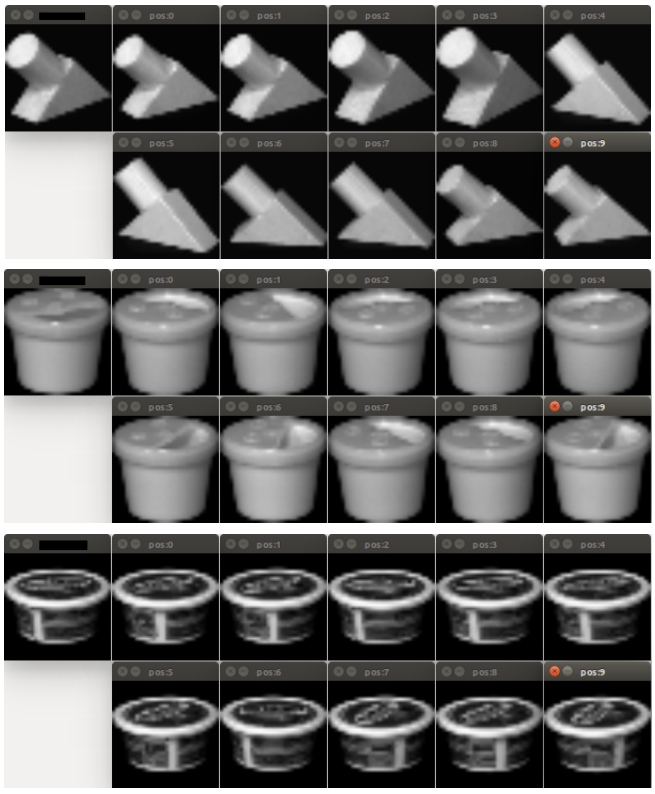


Figura 5.10: Recuperaciones en COIL20 aplicando el *DAE*

5.5.1.2. Pruebas con el Modelo *CNN-DAE*

Para la creación de la CNN se utilizó la siguiente configuración:

- Método *pooling*: *Max-Pooling*.
- Función de activación: *ReLU*.
- Capas convolutivas y *pooling*: 2 Capas.
  - Primera capa convolutiva: 6 *kernels* de tamaño 15x15x1, *pooling*: 2x2
  - Segunda capa convolutiva: 6 *kernels* de tamaño 6x6x1, *pooling*: 2x2
- Capas totalmente conectadas: 2 capas
  - Primera capa totalmente conectada: 512 neuronas, ratio *dropout*: 0.5
  - Segunda capa totalmente conectada: 128 neuronas, ratio *dropout*: 0.5
- Capa de salida:
  - 10 neuronas (10 categorías)
  - Funcion de costo: *Softmax*.

En la Figura 5.11 se puede apreciar la disminución del error dentro de 20000 iteraciones de la CNN, con este entrenamiento se obtiene un 100 % en la clasificación de las imágenes, tanto en el conjunto de entrenamiento como en el de prueba.

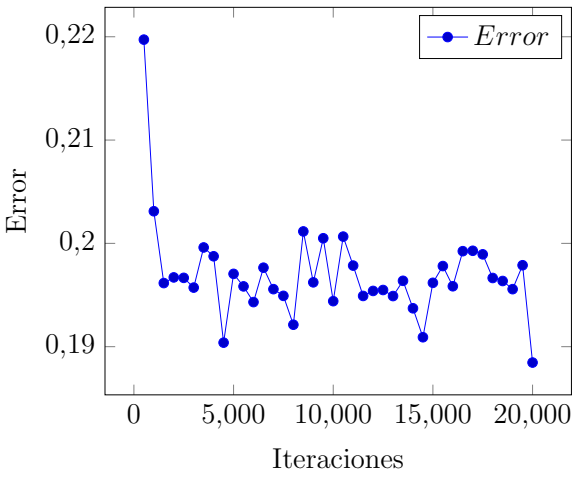


Figura 5.11: Disminución del error en la clasificación de COIL20 con una *CNN*

Luego de la aplicación de la CNN sobre los datos de entrada, se aplicó el DAE para la reducción dimensional y la distribución de los resultados pueden apreciarse en la Figura 5.12

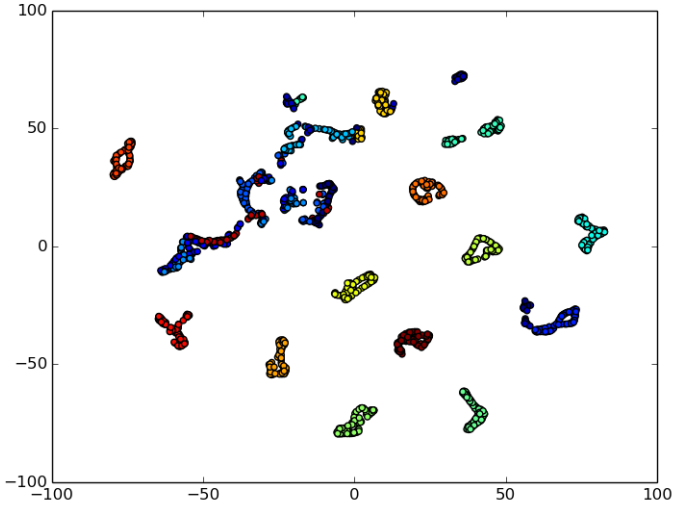


Figura 5.12: Proyección en dos dimensiones de la reducción dimensional en COIL20 con el *CNN-DAE*

La gráfica *precision* y *recall* se muestra en la Figura 5.13 y con los mismos se obtiene un valor para el MAP de 0.75. En la Figura 5.14 se observan algunas de las imágenes recuperadas, la imagen de arriba a la izquierda es la imagen de prueba y las imágenes restantes son las recuperadas ordenadas en el orden: de izquierda a derecha y de arriba hacia abajo.

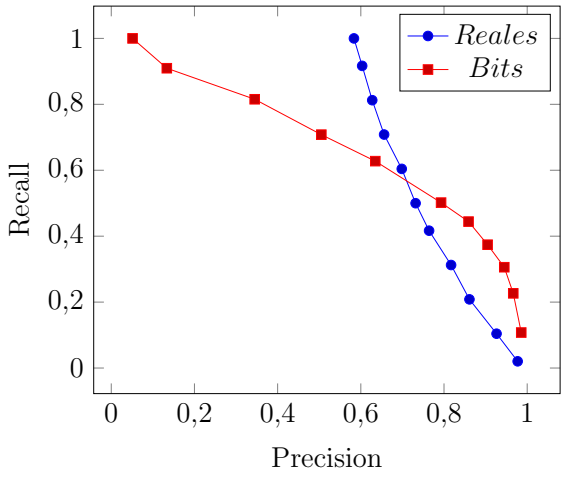


Figura 5.13: Resultados *precision-recall* COIL20 con *CNN-DAE*

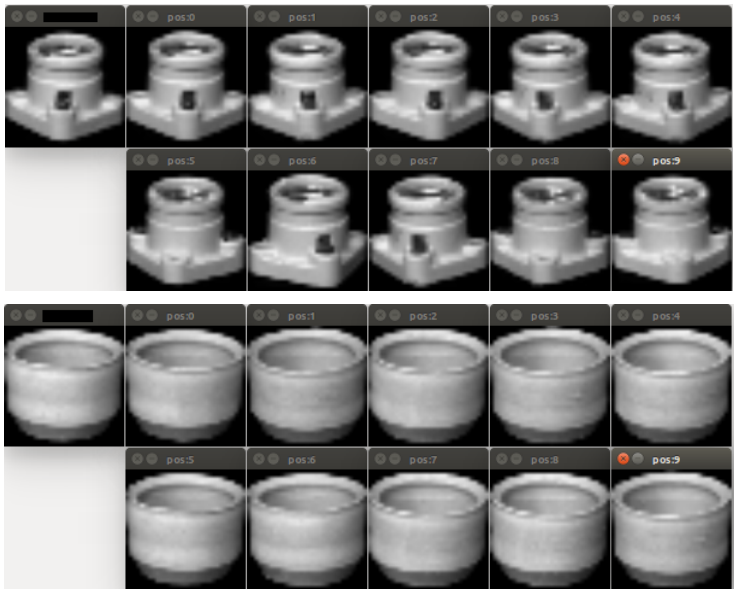


Figura 5.14: Recuperaciones en COIL20 aplicando el *CNN-DAE*

5.5.1.3. Comparación entre los modelos *CNN-DAE* y el *DAE*

Analizando las curvas de *precision-recall* en la Figura 5.15 y las medidas MAP sobre el conjunto de datos COIL20 (67 % en *DAE* y 75 % en *CNN-DAE*) muestran claramente que el modelo *CNN-DAE* obtiene une ventaja sobre el otro.

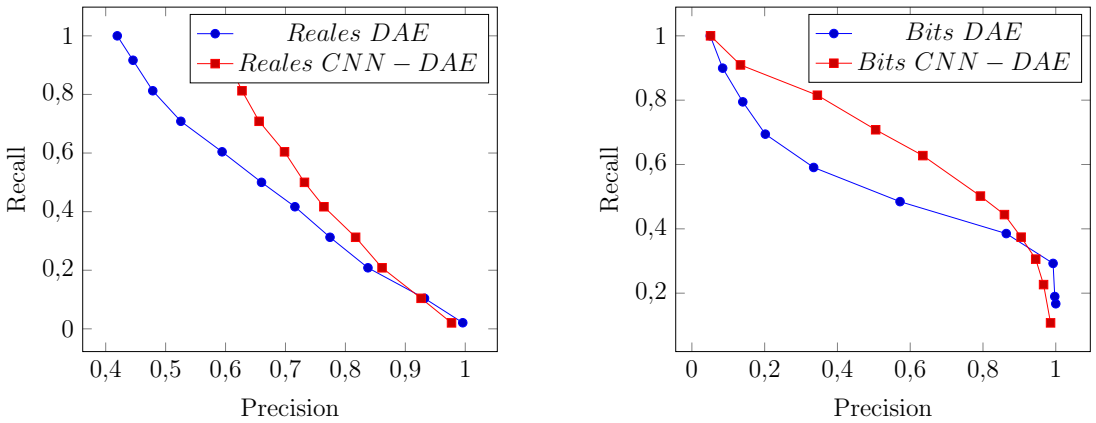


Figura 5.15: Comparación *precision-recall* entre *DAE* y *CNN-DAE*

5.5.2. Resultados en MNIST

5.5.2.1. Pruebas con el DAE

La Figura 5.16 muestra los resultados obtenidos luego de aplicar la reducción dimensional con un DAE sobre el conjunto de datos MNIST. Podemos ver la distribución de los datos logrados con un DAE.

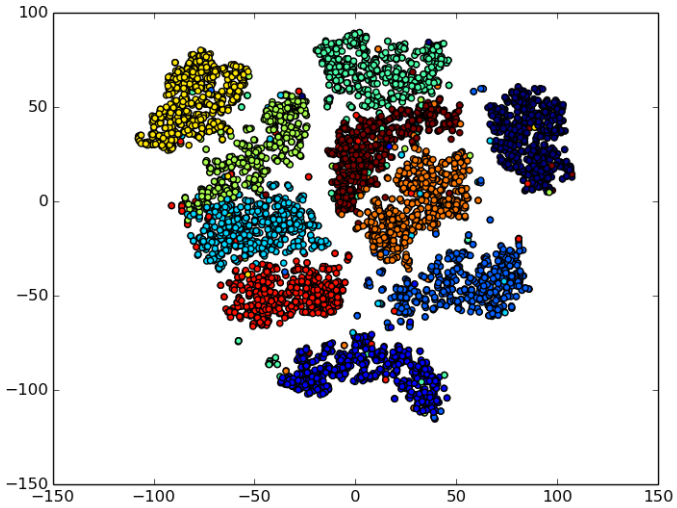


Figura 5.16: Proyección en dos dimensiones de la reducción dimensional en MNIST con el *DAE*

La Figura 5.17 muestra los resultados de *precision-recall* obtenidos, estos resultado logran obtener un una medida MAP de 0.45.

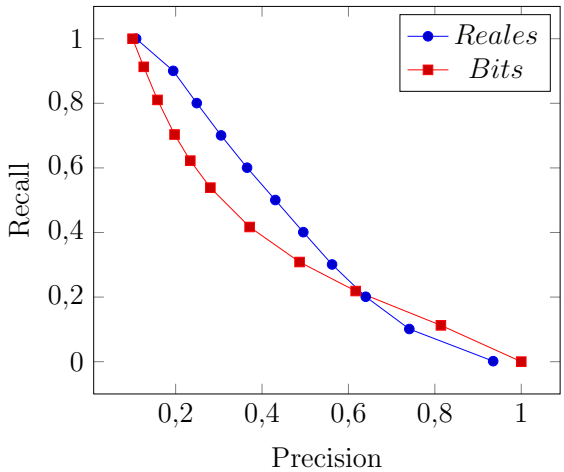


Figura 5.17: Resultados *precision-recall* MNIST con *DAE*

Se aprecia algunas recuperaciones obtenidas con el DAE sobre el conjunto de datos *MNIST* en la Figura 5.18.

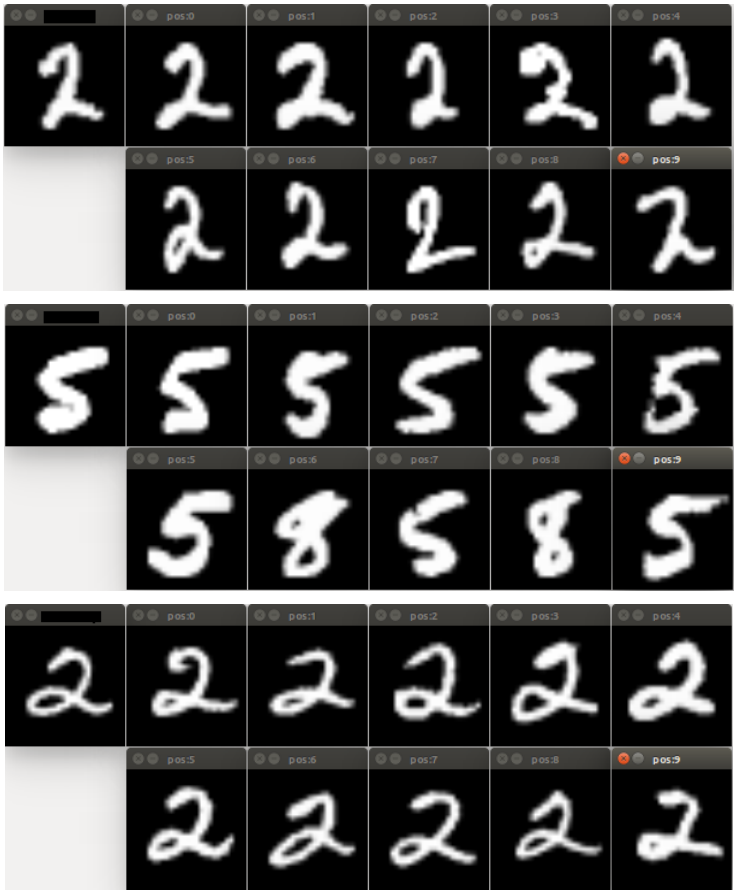


Figura 5.18: Recuperaciones en MNIST aplicando el *DAE*

5.5.2.2. Pruebas con el CNN-DAE

Para la creación de la CNN se utilizó la siguiente configuración:

- Método *pooling*: *Max-Pooling*.
- Función de activación: *ReLU*.
- Capas convolutivas y *pooling*: 2 Capas.
  - Primera capa convolutiva: 6 *kernels* de tamaño 5x5x1, *pooling*: 2x2
  - Segunda capa convolutiva: 6 *kernels* de tamaño 3x3x1, *pooling*: 2x2
- Capas totalmente conectadas: 2 capas
  - Primera capa totalmente conectada: 200 neuronas, ratio *dropout*: 0.5
  - Segunda capa totalmente conectada: 200 neuronas, ratio *dropout*: 0.5
- Capa de salida:
  - 10 neuronas (10 categorías)

- Funcion de costo: *Softmax*.

En la Figura 5.19 se puede apreciar la disminución del error dentro de 20000 iteraciones (100 épocas y 200 iteraciones por época) de la CNN, con este entrenamiento se obtiene un 97.8 % en la clasificación de las imágenes de entrenamiento y un 97.5 % en las de prueba.

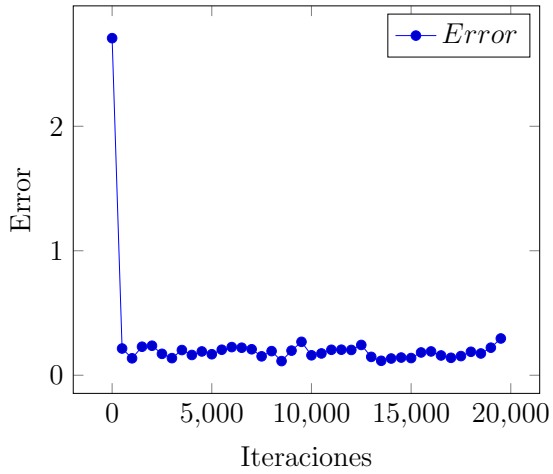


Figura 5.19: Disminución del error en la clasificación del MNIST con una *CNN*

Los resultados luego de la aplicación de la CNN y de la reducción dimensional realizado con un DAE se observan en la figura 5.20.

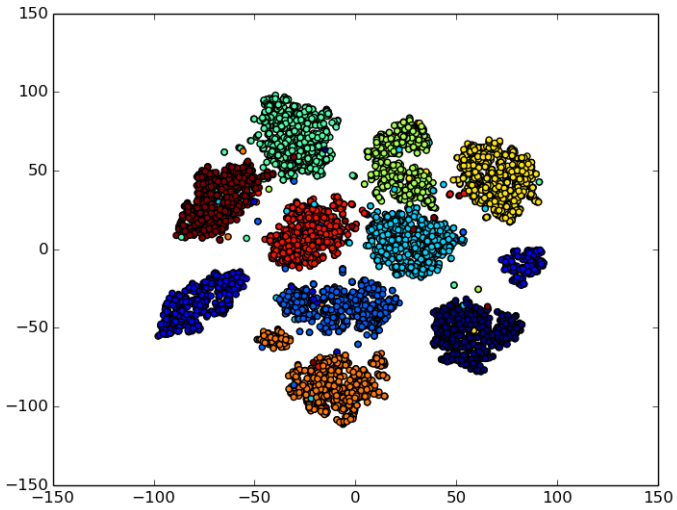


Figura 5.20: Proyección en dos dimensiones de la reducción dimensional en MNIST con el *CNN-DAE*

La correspondiente gráfica *precision-recall* se muestra en la Figura 5.21 con los cuales se obtiene un MAP de 0.56.

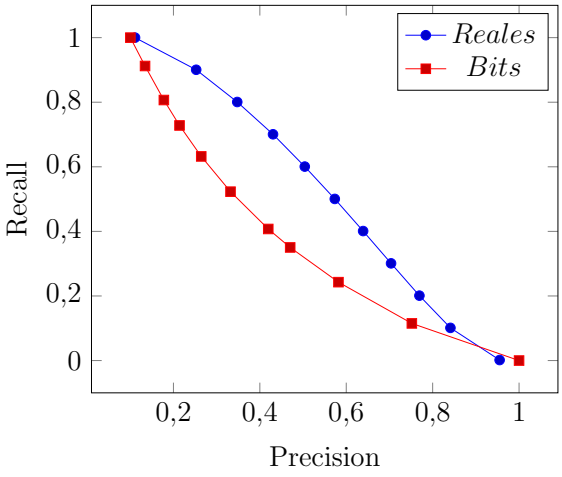


Figura 5.21: Resultados *precision-recall* en MNIST con *CNN-DAE*

Se puede apreciar algunos de los *kernels* aprendidos en el entrenamiento de la CNN en la Figura 5.22. La primera fila muestra los *kernels* de la primera capa convolutiva y en la segunda fila los *kernels* de la segunda capa convolutiva.

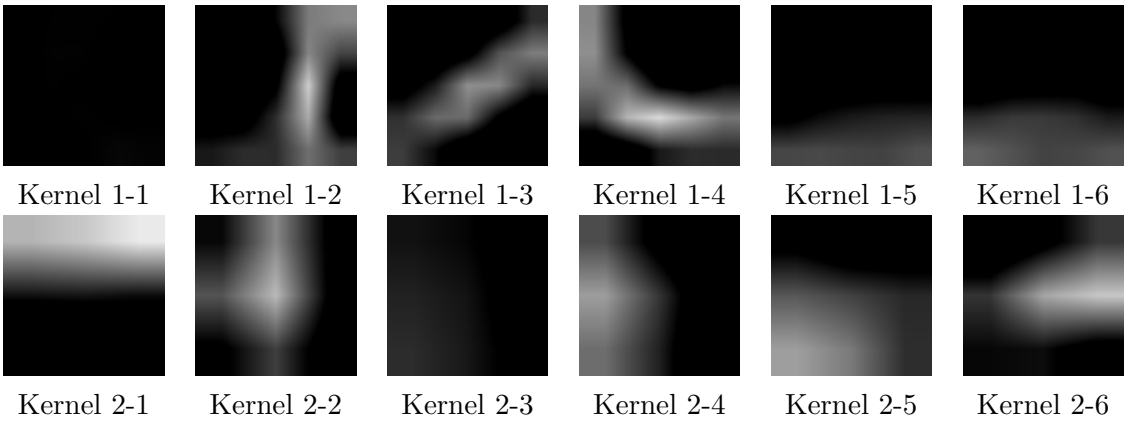


Figura 5.22: *Kernels* al entrenar MNIST con el modelo *CNN-DAE*

Algunas de las reconstrucciones obtenidas en la aplicación del modelo *CNN-DAE* sobre el conjunto de datos MNIST se observan en la Figura 5.23

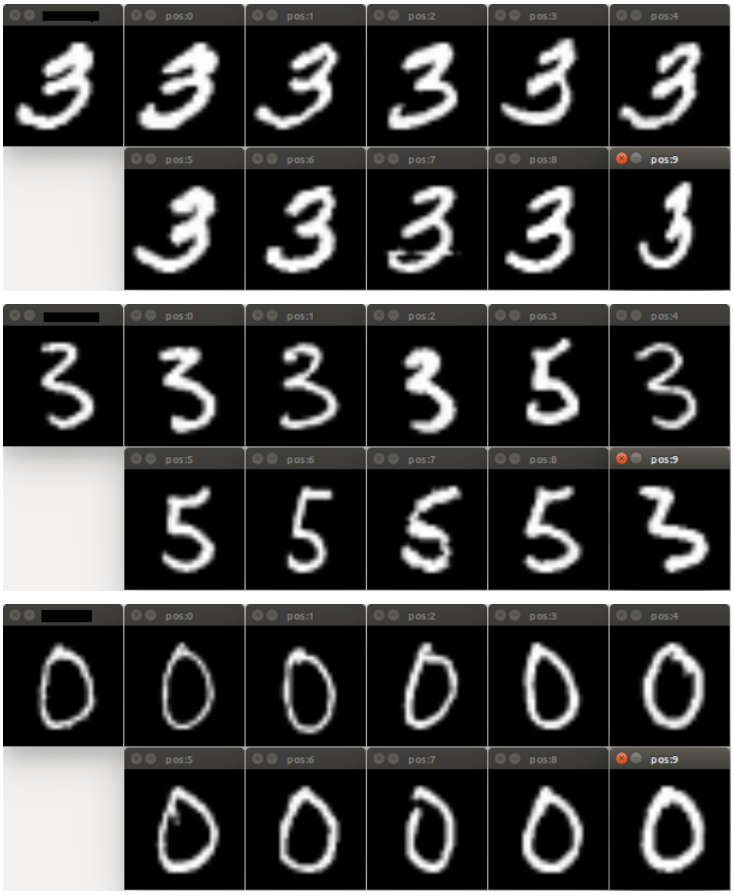


Figura 5.23: Recuperaciones en MNIST aplicando el *CNN-DAE*

5.5.2.3. Comparación entre los modelos *CNN-DAE* y el *DAE*

Analizando las curvas de *precision-recall* en la Figura 5.24 y las medidas MAP sobre el conjunto de datos MNIST (45 % en *DAE* y 56 % en *CNN-DAE*) muestran claramente que el modelo *CNN-DAE* obtiene une ventaja sobre el otro.

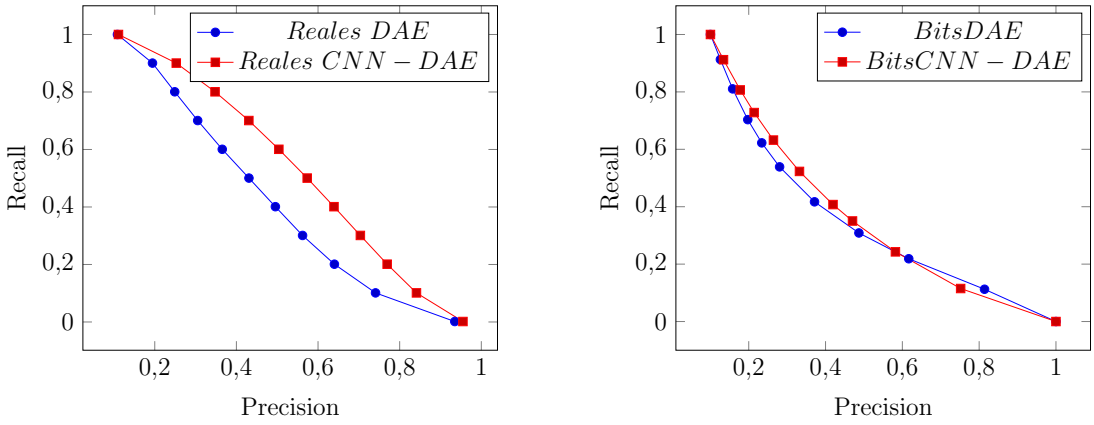


Figura 5.24: Comparación *precision-recall* entre *DAE* y *CNN-DAE*

5.5.3. Resultados en CIFAR10

5.5.3.1. Pruebas con el *DAE*

Los resultados de la reducción dimensional con el DAE se muestran en La Figura 5.25.

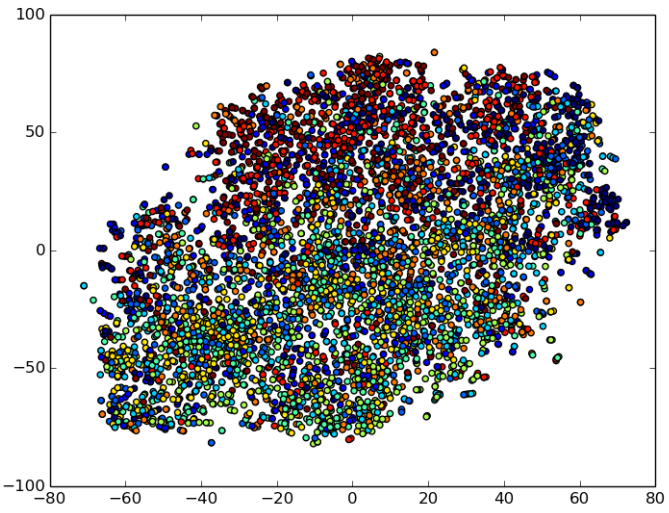


Figura 5.25: Proyección en dos dimensiones de la reducción dimensional en CIFAR10 con el *DAE*

La Figura 5.26 muestra los resultados de *precision-recall* obtenidos, estos resultados logran obtener un una medida MAP de 0.138.

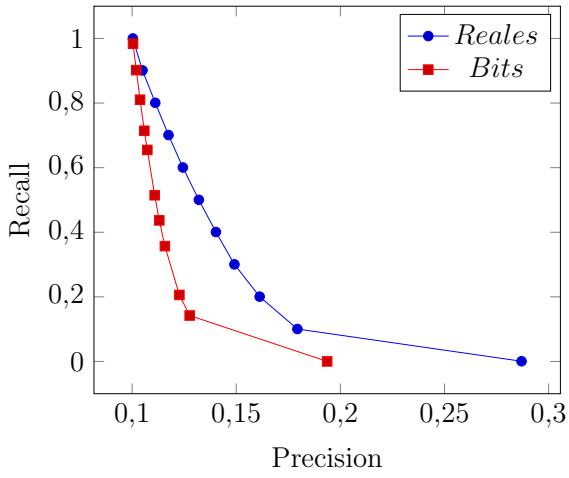


Figura 5.26: Resultados *precision-recall* CIFAR10 con *DAE*

También se puede apreciar algunas recuperaciones obtenidas en la Figura 5.27

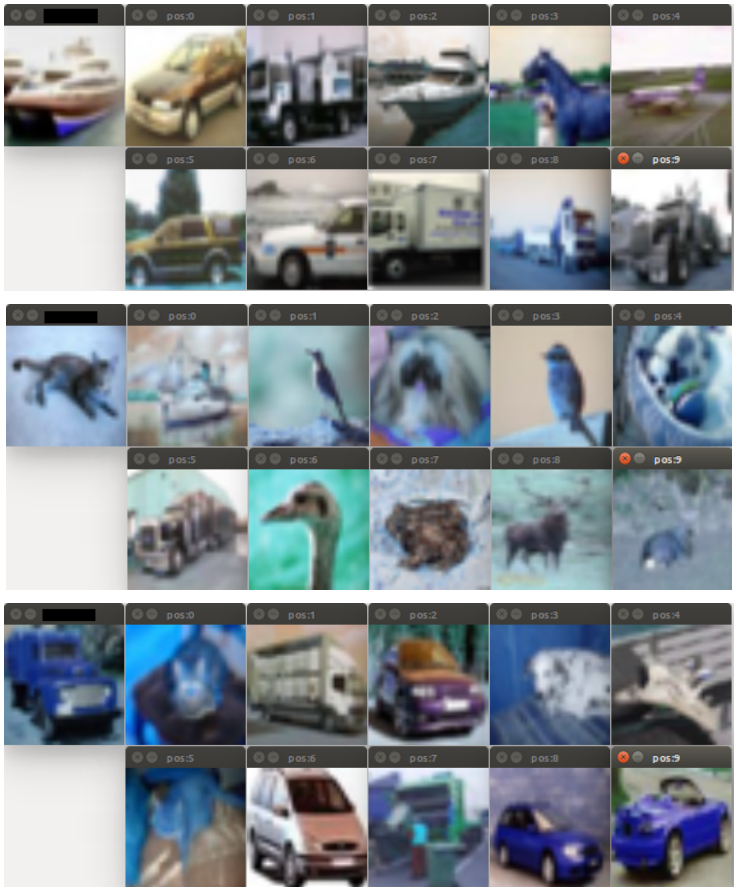


Figura 5.27: Recuperaciones en CIFAR10 aplicando el *DAE*

5.5.3.2. Pruebas con el CNN-DAE

Para la creación de la CNN se utilizó la siguiente configuración:

- Método *pooling*: *Max-Pooling*.
- Función de activación: *ReLU*.
- Capas convolutivas y *pooling*: 2 capas.
  - Primera capa convolutiva: 10 *kernels* de tamaño 15x15x3, *pooling*: 2x2
  - Segunda capa convolutiva: 12 *kernels* de tamaño 6x6x3, *pooling*: 2x2
- Capas totalmente conectadas: 2 capas
  - Primera capa totalmente conectada: 512 neuronas, ratio *dropout*: 0.5
  - Segunda capa totalmente conectada: 128 neuronas, ratio *dropout*: 0.5
- Capa de salida:
  - 10 neuronas (10 categorías)

- Funcion de costo: *Softmax*.

En la Figura 5.28 se puede apreciar la disminución del error dentro de 40000 iteraciones (100 épocas y 400 iteraciones por época) de la CNN, con este entrenamiento se obtiene un 86.9 % en la clasificación de las imágenes de entrenamiento y un 85.8 % en las de prueba.

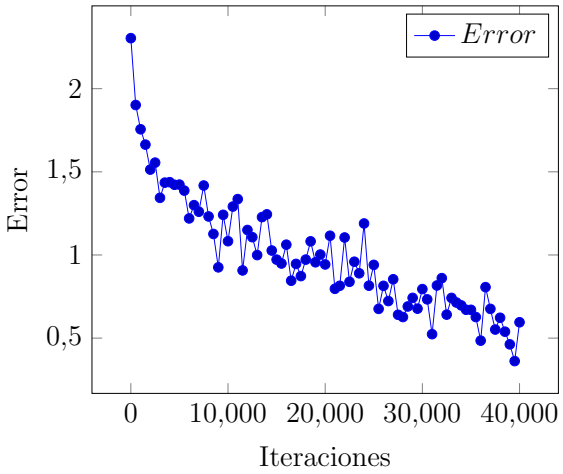


Figura 5.28: Disminución del error en la clasificación del CIFAR10 con una *CNN*

Los resultados luego de la aplicación de la CNN y de la reducción dimensional se observan en la Figura 5.29.

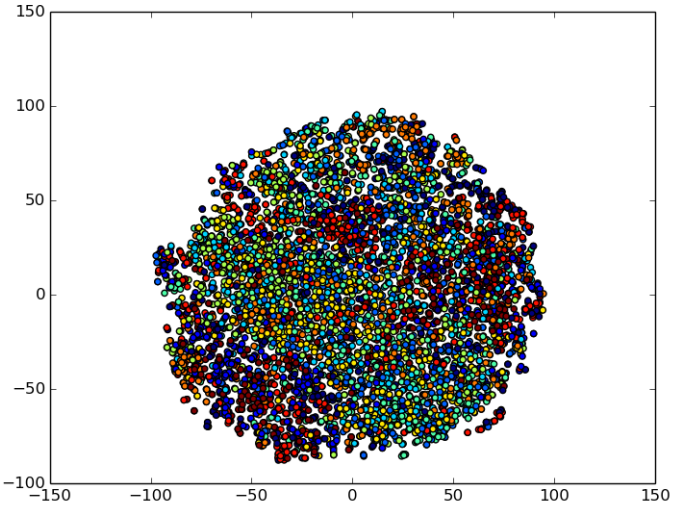


Figura 5.29: Proyección en dos dimensiones de la reducción dimensional en CIFAR10 con el *CNN-DAE*

La correspondiente gráfica *precision-recall* se muestra en la Figura 5.30, los cuales logran un MAP de 0.1442.

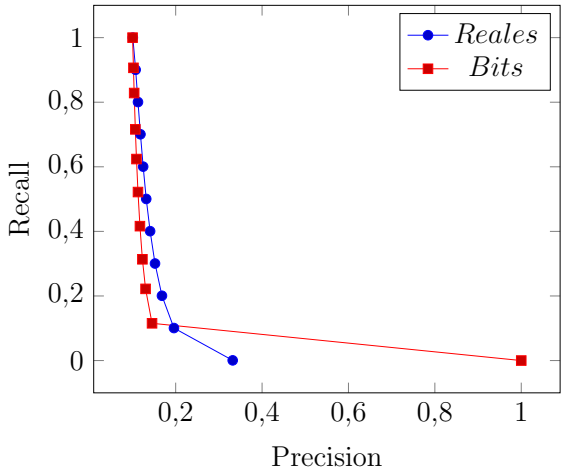


Figura 5.30: Resultados *precision-recall* en CIFAR10 con *CNN-DAE*

Algunos de los *kernels* aprendidos en el entrenamiento de la CNN se encuentran en la Figura 5.31. La primera fila muestra los *kernels* de la primera capa convolutiva y en la segunda fila los *kernels* de la segunda capa convolutiva.

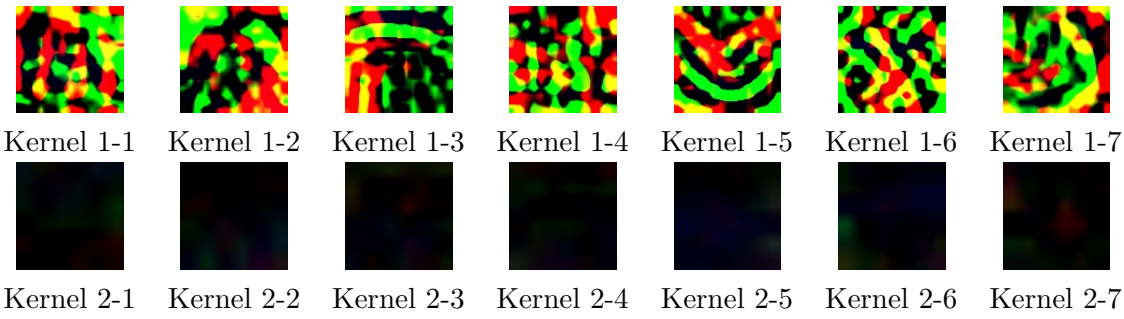


Figura 5.31: *Kernels* al entrenar CIFAR10 con el modelo *CNN-DAE*

Algunas de las reconstrucciones obtenidas en la aplicación del modelo *CNN-DAE* se observan en la Figura 5.32

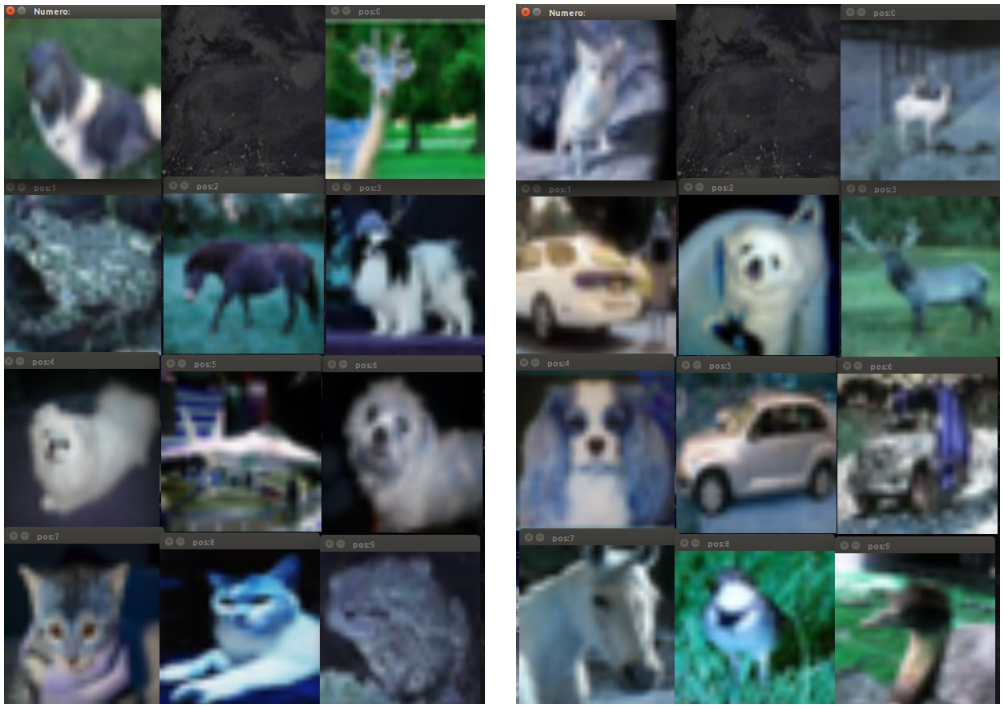


Figura 5.32: Recuperaciones en CIFAR10 aplicando el *CNN-DAE*

5.5.3.3. Comparación entre los modelos *CNN-DAE* y el *DAE*

Analizando las curvas de *precision-recall* en la Figura 5.33 y las medidas MAP sobre el conjunto de datos CIFAR10 (13.8 % en *DAE* y 14.4 % en *CNN-DAE*) muestran que el modelo *CNN-DAE* obtiene une ventaja sobre el otro.

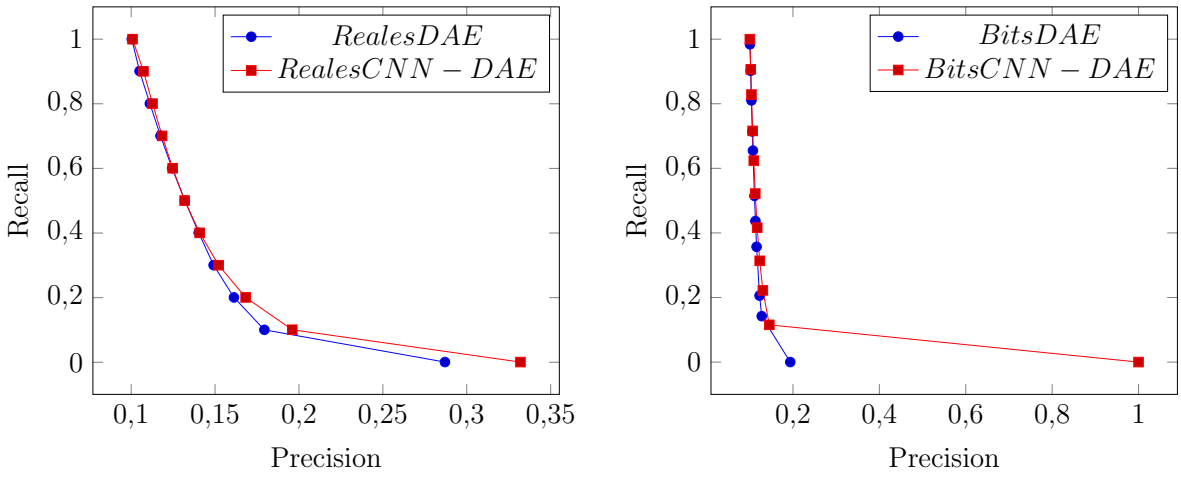


Figura 5.33: Comparación *precision-recall* entre *DAE* y *CNN-DAE*

Como se pudo observar en los resultados obtenidos, el modelo propuesto *CNN-DAE* logra mejorar los resultados al solo aplicar un DAE en la tarea de Recuperación de Imágenes sobre los 3 conjuntos de datos utilizados: COIL20, MNIST, CIFAR10, ver Tabla 5.2. Esta demostración nos da la intuición de que el modelo propuesto pue-

de lograr mejorar los resultados obtenidos en (Wang et al., 2014) en el proceso de recuperación de imágenes.

	MAP DAE	MAP CNN-DAE
COIL20	0.67	0.75
MNIST	0.45	0.56
CIFAR10	0.13	0.14

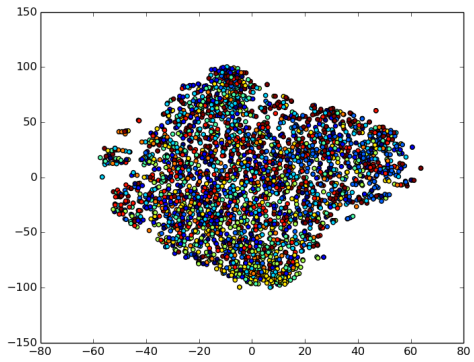
Cuadro 5.2: Comparación de MAP entre *CNN-DAE* y *DAE*

5.6. Resultados de la Recuperación de Información Multi-Modal en el Conjunto de Datos WIKI

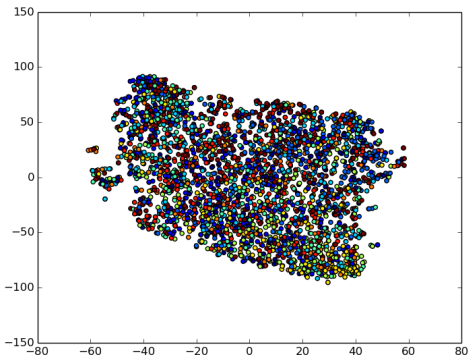
Con el conjunto de datos WIKI, se evaluará el rendimiento del Modelo General propuesto en la tarea de recuperación de información multi-modal. Antes de evaluar el rendimiento del modelo general, se realizarán algunas pruebas que fueron realizadas para mejorar los resultados.

5.6.1. Normalización de los Datos de Entrada al Modelo *CNN-DAE*

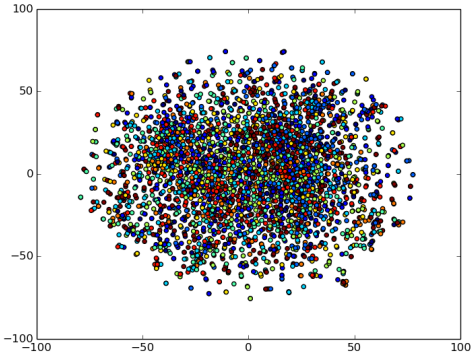
Se realizaron pruebas de normalización de los datos de entrada con tres de las técnicas mas conocidas: *Mean Subtraction*, *Feature Standardization* y *PCA/ZCA Whitening* [13]. Los resultados de las pruebas con estas técnicas se muestran en la Figura 5.34.



Sin Normalización



Mean Subtraction



Feature Standarization con *ZCA Whitening*

Figura 5.34: *Data Normalization* en WIKI aplicando el *CNN-DAE*

[13] http://ufldl.stanford.edu/wiki/index.php/Data_Preprocessing

Se puede apreciar que la diferencia entre no normalizar y aplicar *Mean Subtraction* para normalizar los datos de entrada no es demasiada, pero al aplicar *Feature Standarization* con *ZCA Whitening* los datos tienden a esparcirse o distanciarse. Teniendo en cuenta estos resultados, se aplicará la técnica *Mean Subtraction* sobre los datos de entrada para los modelos propuesto. debido a que se necesita que los datos mantengan su distribución.

5.6.2. Resultados con el Modelo *CNN-DAE*

En la Tabla 5.3 se puede ver el *MAP* y el tamaño de los *Data Codes* utilizados.

Tarea		$Q_{I \to I}$				
Algoritmo		LCMH	CMSSH	CVH	MSAE	CNN-DAE
Dimensión del *Data Code*	16	0.146	0.148	0.147	**0.162**	0.118
	32	0.147	0.149	0.148	**0.162**	0.120
	128	-	-	-	-	**0.121**

Cuadro 5.3: *MAP* en WIKI con el modelo *CNN-DAE*

La sintaxis $Q_{I \to I}$ significa que se realiza una recuperación de tipo *intra-modal* en el espacio de las imágenes. En la tabla anterior se evalúan los resultados obtenidos con diversas investigaciones en el estado del arte de la recuperación de información multi-modal: LCMH (Zhu et al., 2013), CMSSH (Bronstein et al., 2010), CVH (Shaishav y Raghavendra, 2011) y MSAE (Wang et al., 2016). De tal manera como en las investigaciones mencionadas, se probaron distintas configuraciones en el tamaño de los *Data Codes*.

5.6.3. Resultados en el Modelo *D2V-DAE*

En la Tabla 5.4 se puede ver el *MAP* y el tamaño de los *data codes* utilizados.

Tarea		$Q_{T \to T}$				
Algoritmo		LCMH	CMSSH	CVH	MSAE	D2V-DAE
Dimensión del *Data Code*	16	0.359	0.318	0.153	**0.462**	0.342
	32	0.333	0.312	0.152	**0.453**	0.362
	128	-	-	-	-	**0.365**

Cuadro 5.4: *MAP* en WIKI con el modelo *D2V-DAE*

La sintaxis $Q_{T \to T}$ significa que se realiza una recuperación de tipo *intra-modal* en el espacio de los textos. De tal manera como en la evaluación de modelo *CNN-DAE*, se probaron distintas configuraciones en el tamaño de los *data codes*.

5.6.4. Resultados en el Modelo de Recuperación de Información Multi-Modal propuesto

Los resultados se puede apreciar en la siguientes Tablas 5.5 y 5.6.

Tarea		$Q_{I \to T}$				
Algoritmo		LCMH	CMSSH	CVH	MSAE	Modelo Propuesto
Dimensión del *Data Code*	16	0.133	0.138	0.126	**0.182**	0.16
	32	0.137	0.133	0.128	**0.187**	0.169
	128	-	-	-	-	**0.17**

Cuadro 5.5: *MAP* en WIKI con el modelo de recuperacion multi-modal (de imagen a texto)

Tarea		$Q_{T \to I}$				
Algoritmo		LCMH	CMSSH	CVH	MSAE	Modelo Propuesto
Dimensión del *Data Code*	16	0.117	0.140	0.122	**0.179**	0.129
	32	0.119	0.137	0.123	**0.179**	0.13
	128	-	-	-	-	**0.13**

Cuadro 5.6: *MAP* en WIKI con el modelo de recuperación multi-modal (de textos a imagen)

La sintaxis $Q_{I \to T}$ significa que se realiza una recuperación de tipo *cross-modal* ingresando una imagen y recuperando documentos de texto.

La sintaxis $Q_{T \to I}$ significa que se realiza una recuperación de tipo *cross-modal* ingresando un documento de texto y recuperando imágenes.

De los resultados mostrados en esta sección se pueden obtener diversas conclusiones e interpretaciones:

- Se puede observar que los resultados no lograron superar los resultados en el estado del arte alcanzado por el modelo MSAE en (Wang et al., 2016).
- Aunque no se pudo alcanzar los resultado deseados, el modelo propuesto logra posicionarse dentro de los mejores modelos en el proceso de recuperación multi-modal.
- En el caso de la recuperación de imágenes, uno de los motivos por los que se cree que no se pudo alcanzar el Estado del arte, es que las imágenes de entrada fueron escaladas a un tamaño de 32x32. Esto provoca que se pierdan algún tipo de información al momento de la creación del espacio semántico *Intra-Modal* de las imágenes. Este escalamiento de las imágenes se propuso con la finalidad de disminuir el tiempo y el costo computacional al entrenar el modelo *CNN-DAE*.

- En el caso de la recuperación de textos, el proceso *paragraph vector* utilizado para representar los documentos, necesita de una considerable cantidad de información para poder realizar una representación adecuada. Debido a que el conjunto de documentos de texto en el conjunto de datos WIKI poseen la característica de ser semánticamente muy distantes, además de poseer reducidos ejemplos de entrenamiento, se intuye que no se pudo representar correctamente los documentos de entrada, por consiguiente no se pudo alcanzar el estado del arte.

- A pesar de las observaciones antes mencionadas el modelo propuesto logra resultados que son comparables con los resultados en el estado del arte.

5.7. Resultados de la Recuperación de Información Multi-Modal en el Conjunto de Datos COCO

5.7.1. Resultados en el Modelo *CNN-DAE*

En la Tabla 5.7 se puede ver el *MAP* y el tamaño de los *Data Codes* utilizados.

Tarea		$Q_{I \to I}$
Algoritmo		**CNN-DAE**
Dimensión del *Data Code*	16	**0.20**
	32	**0.197**
	128	**0.198**

Cuadro 5.7: *MAP* en COCO con el modelo *CNN-DAE*

5.7.2. Resultados en el Modelo *D2V-DAE*

En la Tabla 5.8 se puede ver el *MAP* y el tamaño de los *Data Codes* utilizados.

Tarea		$Q_{T \to T}$
Algoritmo		**D2V-DAE**
Dimensión del *Data Code*	16	**0.16**
	32	**0.17**
	128	**0.1678**

Cuadro 5.8: *MAP* en COCO con el modelo *D2V-DAE*

5.7.3. Resultados en el Modelo de Recuperación de Información Multi-Modal propuesto

Los resultados se puede apreciar en la siguientes tablas 5.9, 5.10.

Tarea		$Q_{I \to T}$
Algoritmo		Modelo Propuesto
Dimensión del *Data Code*	16	**0.143**
	32	**0.14**
	128	**0.1435**

Cuadro 5.9: *MAP* en COCO con el modelo general - imagen a texto

Tarea		$Q_{T \to I}$
Algoritmo		Modelo Propuesto
Dimensión del *Data Code*	16	**0.165**
	32	**0.16**
	128	**0.1645**

Cuadro 5.10: *MAP* en COCO con el modelo general - texto a imagen

De los resultados mostrados en esta sección se pueden obtener diversas conclusiones e interpretaciones:

- El conjunto de datos COCO posee dos características muy especiales:
  - Las imágenes pueden pertenecer a varias categorías. Por ejemplo, una imagen se compone de varios objetos dentro de la misma, donde cada objeto pertenece a una categoría específica.
  - Cada imagen posee en promedio 5 sentencias de texto, las cuales describen la imagen.
- Esta investigación utiliza este conjunto de datos debido a que cumple con los requisitos necesarios. Posee pares de imágenes y sentencias de texto.
- Debido a las características especiales de este Conjunto de Datos, se debe de realizar un pre-procesamiento especial para poder ser utilizado en la tarea de recuperación de información.
  - Debido a que cada imagen posee varias categorías, es necesario crear un procedimiento para que cada imagen pertenezca a una sola categoría. La selección de la categoría de cada imagen dependerá área de los objetos dentro de cada imagen. Por ejemplo, supongamos que una imagen posee 10 objetos, 5 de los cuales pertenecen a la categoría A, 3 a la categoría B y dos a la categoría C. Ahora, a todos los elementos de la categoría A les corresponde

un 30 % del área global de la imagen, a los elementos de la categoría B les corresponde un 50 % y a los de la categoría C un 20 %. Según las características mencionadas, a esta imagen se le asignará la categoría B, debido a que los elementos de esta categoría poseen una mayor área dentro de la imagen.

- Una vez seleccionada la categoría general de la imagen, se asignará esta misma categoría a cada una de las sentencias de texto que describen la imagen.
- Con este procedimiento lograremos obtener un conjunto de datos que puede ser utilizado en la tarea de recuperación de información multi-modal.

- Como un aporte al trabajo de investigación propuesto, los resultados mostrados en esta sección son los primeros resultados obtenidos en el estado del arte utilizando el conjunto de datos COCO en la tarea de recuperación de información multi-modal.

5.8. Conclusiones del Capítulo

En este capítulo se desarrollaron los modelos propuestos en el trabajo de investigación. El proceso de implementación mostrado en este capítulo esta relacionado con los pasos desarrollados en el capítulo de la Propuesta. Se brindaron muchas anotaciones especiales en el desarrollo de todo el modelo de recuperación de información multi-modal; como también, conclusiones de cada sección dentro del capítulo.

Capítulo 6

Conclusiones y Trabajos Futuros

Las conclusiones del presente trabajo de investigación estarán directamente relacionados con los objetivos planteados en el Capítulo 1:

- Según se pudo apreciar en la literatura, se pueden pre-entrenar los *Deep Autoencoders* utilizando Máquinas de Boltzmann Restringidas o los *Autoencoders*, en esta investigación se evaluó el rendimiento en ambos casos y los resultados demuestran que las RBM's son mejores al pre-entrenar los DAE's.

- En la creación de un modelo que pueda crear vectores representativos de los textos de entrada se utilizó el método *Paragraph Vector*. Este modelo fue entrenado con la finalidad de crear vectores a partir de documentos de texto. Al evaluar la creación de los vectores y la similaridad que tienen los vectores pertenecientes a la misma categoría, se obtuvieron los siguientes resultados: En el conjunto de datos WIKI se obtuvo un grado en la precisión del 42 % y en el conjunto de datos COCO del 36 %. Estos resultados fueron obtenidos por los vectores entes del proceso de reducción dimensional del modelo *D2V-DAE*.

- Las pruebas en la evaluación del rendimiento del modelo *intra-modal* de recuperación de imágenes denominado *CNN-DAE*, se desarrollaron en las Secciones 5.5, 5.6.2 y 5.7.1, en donde se puede apreciar que el modelo propuesto logra un mayor porcentaje MAP comparado con la utilización de un solo DAE. Estas pruebas se realizaron con la finalidad de tener un punto base de comparación con el método de recuperación de imágenes propuesto en (Wang et al., 2016).

- Se logró desarrollar y evaluar el modelo de recuperación de información *cross-modal* con la finalidad de establecer una relación entre ambas modalidades. Los resultados demuestran que si bien no se llegó a alcanzar los resultados del estado del arte, esta investigación propone un novedoso modelo el cual puede compararse con modelos mucho más sofisticados.

6.1. Anotaciones Importantes

En todo el trabajo de investigación se desarrolló un modelo que pueda mejorar el rendimiento de la tarea de recuperación de información multi-modal utilizando modelos de aprendizaje profundo. Una de las grandes diferencias en comparación con anteriores y actuales modelos en el estado del arte, es que esos modelos por lo general, no trabajan con documentos de texto completos y utilizan mayormente las etiquetas de los mismos, o en otras investigaciones tratan de reducir el documento de texto a un conjunto de palabras frecuentes, tratando de encontrar una mejor representación del documento. En esta investigación, para el tratamiento de los textos, se utiliza documentos de texto completos, transformándolos a un vector representativo. Esto es uno de los principales aportes del trabajo de investigación.

También se pudo desarrollar una nueva arquitectura para el tratamiento de las imágenes, la cual como se pudo apreciar en los resultados, logra mejorar básicamente el rendimiento de una de las mayores arquitectura utilizadas en la tarea de recuperación de información en imágenes, como son los DAE's. Los resultados mostrados en la Sección 5.5 muestran esta aseveración.

Como se observa en el Capítulo de 5 se utilizó solo dos conjuntos de datos en la evaluación del modelo en general, el conjunto de datos COCO y el conjunto de datos WIKI, esto debido a que poseen párrafos o documentos de texto en vez de etiquetas.

6.2. Trabajos Futuros

Se continuará con esta investigación con la finalidad de poder superar el estado del arte, actualizando el modelo de CNN utilizado por otros modelos actuales y de mejor rendimiento.

En esta investigación el desarrollo de los modelos se realizó teniendo en cuenta un desarrollo secuencial, esto quiere decir que no se utilizaron las Unidades Procesadoras de Gráficos o *GPU*'s o algún método de paralelización de procesos, como consecuencia el tiempo computacional fue mucho mayor al esperado. Con algunos conjuntos de datos como el CIFAR10 o el COCO el entrenamiento demoró en promedio 4 días. Por lo cual, se utilizará programación paralela para mejorar el tiempo computacional de los modelos.

Bibliografía

Baeza, Y., Ricardo, et al. (2011). Addison-Wesley Publishing Company, USA, 2nd edition.

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127. Also published as a book. Now Publishers, 2009.

Bengio, Y., Ducharme, R., et al. (2003). A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155.

Bertona, L. (2005). Entrenamiento de redes neuronales basado en algoritmos evolutivos.

Bronstein, M. M., Bronstein, A. M., et al. (2010). Data fusion through cross-modality metric learning using similarity-sensitive hashing. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3594–3601.

Campos, R., Dias, G., et al. (2014). Survey of temporal information retrieval and related applications. *ACM Comput. Surv.*, 47(2):15:1–15:41.

Cauwenberghs, G. (1993). A fast stochastic error-descent algorithm for supervised learning and optimization. In *Advances in Neural Information Processing Systems 5, [NIPS Conference]*, pages 244–251, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Cho, K., van Merriënboer, B., et al. (2014). Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

De Falco, I., Della Cioppa, A., et al. (1998). Artificial neural networks optimization by means of evolutionary algorithms. In *Soft Computing in Engineering Design and Manufacturing*, pages 3–12. Springer.

Deng, L., He, X., et al. (2013). Deep stacking networks for information retrieval. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 3153–3157. IEEE.

Deng, L. y Yu, D. (2014). Deep learning: Methods and applications. *Found. Trends Signal Process.*, 7(3-4):197–387.

Deng, L., Yu, D., et al. (2012). Scalable stacking and learning for building deep architectures. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 2133–2136.

Hinton, G. (2010). A practical guide to training restricted boltzmann machines. *Momentum*, 9(1):926.

Hinton, G. E. y Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

Hornik, K., Stinchcombe, M., et al. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366.

Huang, P.-S., He, X., et al. (2013). Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22Nd ACM International Conference on Conference on Information & Knowledge Management*, CIKM '13, pages 2333–2338, New York, NY, USA. ACM.

Hutchinson, B., Deng, L., et al. (2013). Tensor deep stacking networks. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1944–1957.

Ian Goodfellow, Y. B. y Courville, A. (2016). Deep learning. Book in preparation for MIT Press.

Kanhabua, N. (2012). *Time-aware Approaches to Information Retrieval*. PhD thesis, Norwegian University of Science and Technology Faculty of Information Technology, Mathematics and Electrical Engineering. Department of Computer and Information Science.

Kopliku, A., Pinel-Sauvagnat, K., et al. (2014). Aggregated search: A new information retrieval paradigm. *ACM Comput. Surv.*, 46(3):41:1–41:31.

Krizhevsky, A., Sutskever, I., et al. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Le, Q. V. y Mikolov, T. (2014). Distributed representations of sentences and documents. *CoRR*, abs/1405.4053.

LeCun, Y., Bottou, L., et al. (1998). Gradient-based learning applied to document recognition. 86(11):2278–2324.

Maaten, L. v. d. y Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605.

Manning, C. D., Raghavan, P., et al. (2009). *Introduction to Information Retrieval*. Cambridge University Press.

Markov, I. (2014). *Uncertainty in Distributed Information Retrieval*. PhD thesis, Faculty of Informatics of the Università della Svizzera Italiana.

McCulloch, W. S. y Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.

Mikolov, T., Chen, K., et al. (2013a). Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Mikolov, T., Kopecky, J., et al. (2009). Neural network based language models for highly inflective languages. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4725–4728.

Mikolov, T., Sutskever, I., et al. (2013b). Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546.

Minsky, M. y Papert, S. (1969). Perceptron (expanded edition).

Morin, F. y Bengio, Y. (2005). Hierarchical probabilistic neural network language model. In *AISTATS'05*, pages 246–252.

Ngiam, J., Khosla, A., et al. (2011). Multimodal deep learning. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 689–696.

Olabe, X. B. (1998). Redes neuronales artificiales y sus aplicaciones.

Pérez, S. (2015). *Metodologías de Diseño de Redes Neuronales sobre Dispositivos Digitales Programables para Procesado de Señales en Tiempo Real*. PhD thesis, Departamento de Señales y Comunicaciones de la Universidad de las Palmas de Gran Canaria.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

Rumelhart, D. E., Hinton, G. E., et al. (1988). Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1.

Salakhutdinov, R. y Hinton, G. (2009). Semantic hashing. *Int. J. Approx. Reasoning*, 50(7):969–978.

Shaishav, K. y Raghavendra, U. (2011). Learning hash functions for cross-view similarity search. IJCAI.

Shen, Y., He, X., et al. (2014). Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the Companion Publication of the 23rd International Conference on World Wide Web Companion*, WWW Companion '14, pages 373–374, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Srivastava, N. y Salakhutdinov, R. (2014). Multimodal learning with deep boltzmann machines. *Journal of Machine Learning Research*, 15:2949–2980.

Strotgen, J. (2015). *Domain-sensitive Temporal Tagging for Event-centric Information Retrieval*. PhD thesis, Institute of Computer Science, Ruprecht-Karls-University Heidelberg.

Szegedy, C., Liu, W., et al. (2014). Going deeper with convolutions. *CoRR*, abs/1409.4842.

Szegedy, C., Vanhoucke, V., et al. (2015). Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567.

Turian, J., Ratinov, L., et al. (2010). Word representations: a simple and general method for semi-supervised learning.

Van Der Maaten, L. (2014). Accelerating t-sne using tree-based algorithms. *Journal of machine learning research*, 15(1):3221–3245.

Vikram, A. (2015). Content-based image retrieval using deep learning. Master's thesis, Rochester Institute of Technology.

Wan, J., Wang, D., et al. (2014). Deep learning for content-based image retrieval: A comprehensive study. In *Proceedings of the 22Nd ACM International Conference on Multimedia*, MM '14, pages 157–166, New York, NY, USA. ACM.

Wang, W., Ooi, B. C., et al. (2014). Effective multi-modal retrieval based on stacked auto-encoders. *Proceedings of the VLDB Endowment*, 7(8):649–660.

Wang, W., Yang, X., et al. (2016). Effective deep learning-based multi-modal retrieval. *The VLDB Journal*, 25(1):79–101.

Widrow, B., Hoff, M. E., et al. (1960). Adaptive switching circuits. In *IRE WESCON convention record*, volume 4, pages 96–104. New York.

Zhu, X., Huang, Z., et al. (2013). Linear cross-modal hashing for efficient multimedia search. In *Proceedings of the 21st ACM International Conference on Multimedia*, MM '13, pages 143–152, New York, NY, USA. ACM.